

general_ledger

Generated by Doxygen 1.8.1.2

Sat Jun 7 2014 18:59:01

Contents

1	General Ledger.	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	9
4.1	ds_list Struct Reference	9
4.1.1	Detailed Description	9
4.1.2	Field Documentation	9
4.1.2.1	current	10
4.1.2.2	data_destructor	10
4.1.2.3	free_on_delete	10
4.1.2.4	head	10
4.1.2.5	length	10
4.1.2.6	tail	10
4.2	ds_list_element Struct Reference	10
4.2.1	Detailed Description	10
4.2.2	Field Documentation	11
4.2.2.1	data	11
4.2.2.2	next	11
4.2.2.3	previous	11
4.3	ds_map Struct Reference	11
4.3.1	Detailed Description	11
4.3.2	Field Documentation	12
4.3.2.1	hash_size	12
4.3.2.2	lists	12
4.4	ds_map_str Struct Reference	12
4.4.1	Detailed Description	12
4.4.2	Field Documentation	12

4.4.2.1	hash_size	13
4.4.2.2	lists	13
4.5	ds_record Struct Reference	13
4.5.1	Detailed Description	13
4.5.2	Field Documentation	13
4.5.2.1	fields	13
4.6	ds_recordset Struct Reference	14
4.6.1	Detailed Description	14
4.6.2	Field Documentation	14
4.6.2.1	field_lengths	14
4.6.2.2	headers	14
4.6.2.3	num_fields	14
4.6.2.4	records	15
4.7	ds_str Struct Reference	15
4.7.1	Detailed Description	15
4.7.2	Field Documentation	15
4.7.2.1	capacity	15
4.7.2.2	data	15
4.7.2.3	length	15
4.8	ds_vector Struct Reference	15
4.8.1	Detailed Description	15
4.8.2	Field Documentation	16
4.8.2.1	current	16
4.8.2.2	data	16
4.8.2.3	data_destructor	16
4.8.2.4	free_on_delete	16
4.8.2.5	size	16
4.9	kv_pair_node Struct Reference	16
4.9.1	Detailed Description	17
4.9.2	Field Documentation	17
4.9.2.1	key	17
4.9.2.2	key	17
4.9.2.3	next	17
4.9.2.4	value	17
4.9.2.5	value	17
4.10	params Struct Reference	17
4.10.1	Detailed Description	18
4.10.2	Field Documentation	18
4.10.2.1	create	18
4.10.2.2	database	18

4.10.2.3	delete_data	18
4.10.2.4	help	19
4.10.2.5	hostname	19
4.10.2.6	list_entities	19
4.10.2.7	list_users	19
4.10.2.8	password	19
4.10.2.9	sample	19
4.10.2.10	username	19
4.10.2.11	version	19
5	File Documentation	21
5.1	config.c File Reference	21
5.1.1	Detailed Description	22
5.1.2	Macro Definition Documentation	22
5.1.2.1	_XOPEN_SOURCE	22
5.1.3	Function Documentation	22
5.1.3.1	get_cmdline_options	22
5.1.3.2	get_configuration	22
5.1.3.3	params_free	23
5.1.3.4	params_init	23
5.2	config.h File Reference	23
5.2.1	Detailed Description	24
5.2.2	Function Documentation	24
5.2.2.1	get_cmdline_options	24
5.2.2.2	get_configuration	25
5.2.2.3	params_free	25
5.2.2.4	params_init	25
5.3	lib/database/database.h File Reference	25
5.3.1	Detailed Description	26
5.4	lib/database/db_connection.h File Reference	27
5.4.1	Detailed Description	27
5.4.2	Function Documentation	28
5.4.2.1	db_connect	28
5.5	lib/database/db_entities.c File Reference	28
5.5.1	Detailed Description	29
5.5.2	Function Documentation	29
5.5.2.1	db_create_entities_table	29
5.5.2.2	db_drop_entities_table	29
5.5.2.3	db_list_entities_report	29
5.6	lib/database/db_entities.h File Reference	29

5.6.1	Detailed Description	30
5.6.2	Function Documentation	31
5.6.2.1	db_create_entities_table	31
5.6.2.2	db_drop_entities_table	31
5.6.2.3	db_list_entities_report	31
5.7	lib/database/db_internal.h File Reference	31
5.7.1	Detailed Description	32
5.8	lib/database/db_query.h File Reference	32
5.8.1	Detailed Description	33
5.8.2	Function Documentation	33
5.8.2.1	db_execute_query	33
5.9	lib/database/db_reporting.c File Reference	33
5.9.1	Detailed Description	34
5.9.2	Function Documentation	34
5.9.2.1	db_create_report_from_query	34
5.10	lib/database/db_reporting.h File Reference	35
5.10.1	Detailed Description	35
5.10.2	Function Documentation	35
5.10.2.1	db_create_recordset_from_query	35
5.10.2.2	db_create_report_from_query	35
5.11	lib/database/db_sampledata.c File Reference	36
5.11.1	Detailed Description	36
5.12	lib/database/db_sampledata.h File Reference	37
5.12.1	Detailed Description	37
5.13	lib/database/db_sql.h File Reference	37
5.13.1	Detailed Description	38
5.13.2	Function Documentation	38
5.13.2.1	db_create_entities_table_sql	38
5.13.2.2	db_create_users_table_sql	39
5.13.2.3	db_drop_entities_table_sql	39
5.13.2.4	db_drop_users_table_sql	39
5.13.2.5	db_list_entities_report_sql	39
5.13.2.6	db_list_users_report_sql	39
5.14	lib/database/db_structure.c File Reference	39
5.14.1	Detailed Description	40
5.14.2	Function Documentation	40
5.14.2.1	db_create_database_structure	40
5.14.2.2	db_delete_database_structure	41
5.15	lib/database/db_structure.h File Reference	41
5.15.1	Detailed Description	41

5.15.2	Function Documentation	42
5.15.2.1	db_create_database_structure	42
5.15.2.2	db_delete_database_structure	42
5.16	lib/database/db_users.c File Reference	42
5.16.1	Detailed Description	43
5.16.2	Function Documentation	43
5.16.2.1	db_create_users_table	43
5.16.2.2	db_drop_users_table	43
5.16.2.3	db_list_users_report	43
5.17	lib/database/db_users.h File Reference	44
5.17.1	Detailed Description	44
5.17.2	Function Documentation	45
5.17.2.1	db_create_users_table	45
5.17.2.2	db_drop_users_table	45
5.17.2.3	db_list_users_report	45
5.18	lib/database/dummy/db_dummy_create_entities_table_sql.c File Reference	45
5.18.1	Detailed Description	45
5.18.2	Function Documentation	46
5.18.2.1	db_create_entities_table_sql	46
5.19	lib/database/dummy/db_dummy_create_users_table_sql.c File Reference	46
5.19.1	Detailed Description	46
5.19.2	Function Documentation	46
5.19.2.1	db_create_users_table_sql	46
5.20	lib/database/dummy/db_dummy_drop_entities_table_sql.c File Reference	46
5.20.1	Detailed Description	47
5.20.2	Function Documentation	47
5.20.2.1	db_drop_entities_table_sql	47
5.21	lib/database/dummy/db_dummy_drop_users_table_sql.c File Reference	47
5.21.1	Detailed Description	47
5.21.2	Function Documentation	47
5.21.2.1	db_drop_users_table_sql	47
5.22	lib/database/dummy/db_dummy_general.c File Reference	48
5.22.1	Detailed Description	48
5.22.2	Macro Definition Documentation	49
5.22.2.1	_XOPEN_SOURCE	49
5.22.3	Function Documentation	49
5.22.3.1	db_connect	49
5.22.3.2	db_create_recordset_from_query	49
5.22.3.3	db_execute_query	49
5.23	lib/database/dummy/db_dummy_list_entities_report_sql.c File Reference	50

5.23.1 Detailed Description	50
5.23.2 Function Documentation	50
5.23.2.1 db_list_entities_report_sql	50
5.24 lib/database/dummy/db_dummy_list_users_report_sql.c File Reference	50
5.24.1 Detailed Description	50
5.24.2 Function Documentation	51
5.24.2.1 db_list_users_report_sql	51
5.25 lib/database/mysql/db_mysql_create_entities_table_sql.c File Reference	51
5.25.1 Detailed Description	51
5.25.2 Function Documentation	51
5.25.2.1 db_create_entities_table_sql	51
5.26 lib/database/mysql/db_mysql_create_users_table_sql.c File Reference	51
5.26.1 Detailed Description	52
5.26.2 Function Documentation	52
5.26.2.1 db_create_users_table_sql	52
5.27 lib/database/mysql/db_mysql_drop_entities_table_sql.c File Reference	52
5.27.1 Detailed Description	52
5.27.2 Function Documentation	53
5.27.2.1 db_drop_entities_table_sql	53
5.28 lib/database/mysql/db_mysql_drop_users_table_sql.c File Reference	53
5.28.1 Detailed Description	53
5.28.2 Function Documentation	53
5.28.2.1 db_drop_users_table_sql	53
5.29 lib/database/mysql/db_mysql_general.c File Reference	53
5.29.1 Detailed Description	54
5.29.2 Function Documentation	55
5.29.2.1 db_connect	55
5.29.2.2 db_create_recordset_from_query	55
5.29.2.3 db_execute_query	55
5.29.3 Variable Documentation	55
5.29.3.1 conn_mss	55
5.29.3.2 main_mss	55
5.30 lib/database/mysql/db_mysql_list_entities_report_sql.c File Reference	56
5.30.1 Detailed Description	56
5.30.2 Function Documentation	56
5.30.2.1 db_list_entities_report_sql	56
5.31 lib/database/mysql/db_mysql_list_users_report_sql.c File Reference	56
5.31.1 Detailed Description	56
5.31.2 Function Documentation	57
5.31.2.1 db_list_users_report_sql	57

5.32 lib/datastruct/data_structures.h File Reference	57
5.32.1 Detailed Description	58
5.33 lib/datastruct/ds_list.c File Reference	58
5.33.1 Detailed Description	59
5.33.2 Function Documentation	59
5.33.2.1 ds_list_append	59
5.33.2.2 ds_list_create	60
5.33.2.3 ds_list_destroy	60
5.33.2.4 ds_list_destructor	60
5.33.2.5 ds_list_element	60
5.33.2.6 ds_list_get_next_data	60
5.33.2.7 ds_list_get_prev_data	61
5.33.2.8 ds_list_is_empty	61
5.33.2.9 ds_list_length	61
5.33.2.10 ds_list_remove_all	61
5.33.2.11 ds_list_remove_tail	62
5.33.2.12 ds_list_seek_end	62
5.33.2.13 ds_list_seek_start	62
5.34 lib/datastruct/ds_list.h File Reference	62
5.34.1 Detailed Description	63
5.34.2 Typedef Documentation	64
5.34.2.1 ds_list	64
5.34.3 Function Documentation	64
5.34.3.1 ds_list_append	64
5.34.3.2 ds_list_create	64
5.34.3.3 ds_list_destroy	64
5.34.3.4 ds_list_destructor	65
5.34.3.5 ds_list_element	65
5.34.3.6 ds_list_get_next_data	65
5.34.3.7 ds_list_get_prev_data	65
5.34.3.8 ds_list_is_empty	66
5.34.3.9 ds_list_length	66
5.34.3.10 ds_list_remove_all	66
5.34.3.11 ds_list_remove_tail	66
5.34.3.12 ds_list_seek_end	66
5.34.3.13 ds_list_seek_start	66
5.35 lib/datastruct/ds_map.c File Reference	67
5.35.1 Detailed Description	68
5.35.2 Function Documentation	68
5.35.2.1 ds_map_destroy	68

5.35.2.2	ds_map_get_value	68
5.35.2.3	ds_map_init	68
5.35.2.4	ds_map_insert	69
5.35.2.5	ds_map_print_all	69
5.36	lib/datastruct/ds_map.h File Reference	69
5.36.1	Detailed Description	70
5.36.2	Typedef Documentation	70
5.36.2.1	ds_map	70
5.36.3	Function Documentation	70
5.36.3.1	ds_map_destroy	70
5.36.3.2	ds_map_get_value	71
5.36.3.3	ds_map_init	71
5.36.3.4	ds_map_insert	71
5.36.3.5	ds_map_print_all	71
5.37	lib/datastruct/ds_map_str.c File Reference	72
5.37.1	Detailed Description	72
5.37.2	Function Documentation	73
5.37.2.1	ds_map_str_destroy	73
5.37.2.2	ds_map_str_get_value	73
5.37.2.3	ds_map_str_init	73
5.37.2.4	ds_map_str_insert	73
5.38	lib/datastruct/ds_map_str.h File Reference	74
5.38.1	Detailed Description	75
5.38.2	Typedef Documentation	75
5.38.2.1	ds_map_str	75
5.38.3	Function Documentation	75
5.38.3.1	ds_map_str_destroy	75
5.38.3.2	ds_map_str_get_value	75
5.38.3.3	ds_map_str_init	75
5.38.3.4	ds_map_str_insert	76
5.39	lib/datastruct/ds_record.c File Reference	76
5.39.1	Detailed Description	77
5.39.2	Function Documentation	77
5.39.2.1	ds_record_clear	77
5.39.2.2	ds_record_create	77
5.39.2.3	ds_record_destroy	78
5.39.2.4	ds_record_destructor	78
5.39.2.5	ds_record_get_field	78
5.39.2.6	ds_record_get_next_data	78
5.39.2.7	ds_record_make_delim_string	78

5.39.2.8	ds_record_make_values_string	79
5.39.2.9	ds_record_seek_start	79
5.39.2.10	ds_record_set_field	79
5.39.2.11	ds_record_size	79
5.39.2.12	ds_record_tokenize	80
5.40	lib/datastruct/ds_record.h File Reference	80
5.40.1	Detailed Description	81
5.40.2	Typedef Documentation	81
5.40.2.1	ds_record	81
5.40.3	Function Documentation	81
5.40.3.1	ds_record_clear	81
5.40.3.2	ds_record_create	82
5.40.3.3	ds_record_destroy	82
5.40.3.4	ds_record_destructor	82
5.40.3.5	ds_record_get_field	82
5.40.3.6	ds_record_get_next_data	82
5.40.3.7	ds_record_make_delim_string	83
5.40.3.8	ds_record_make_values_string	83
5.40.3.9	ds_record_seek_start	83
5.40.3.10	ds_record_set_field	83
5.40.3.11	ds_record_size	84
5.40.3.12	ds_record_tokenize	84
5.41	lib/datastruct/ds_recordset.c File Reference	84
5.41.1	Detailed Description	85
5.41.2	Function Documentation	85
5.41.2.1	ds_recordset_add_record	85
5.41.2.2	ds_recordset_create	86
5.41.2.3	ds_recordset_destroy	86
5.41.2.4	ds_recordset_get_next_insert_query	86
5.41.2.5	ds_recordset_get_text_report	86
5.41.2.6	ds_recordset_next_record	86
5.41.2.7	ds_recordset_num_fields	87
5.41.2.8	ds_recordset_num_records	87
5.41.2.9	ds_recordset_seek_start	87
5.41.2.10	ds_recordset_set_headers	87
5.42	lib/datastruct/ds_recordset.h File Reference	88
5.42.1	Detailed Description	89
5.42.2	Typedef Documentation	89
5.42.2.1	ds_recordset	89
5.42.3	Function Documentation	89

5.42.3.1	ds_recordset_add_record	89
5.42.3.2	ds_recordset_create	90
5.42.3.3	ds_recordset_destroy	90
5.42.3.4	ds_recordset_get_next_insert_query	90
5.42.3.5	ds_recordset_get_text_report	90
5.42.3.6	ds_recordset_next_record	90
5.42.3.7	ds_recordset_num_fields	91
5.42.3.8	ds_recordset_num_records	91
5.42.3.9	ds_recordset_seek_start	91
5.42.3.10	ds_recordset_set_headers	91
5.43	lib/datastruct/ds_str.c File Reference	92
5.43.1	Detailed Description	94
5.43.2	Function Documentation	94
5.43.2.1	ds_str_assign	94
5.43.2.2	ds_str_assign_cstr	94
5.43.2.3	ds_str_char_at_index	94
5.43.2.4	ds_str_clear	95
5.43.2.5	ds_str_compare	95
5.43.2.6	ds_str_compare_cstr	95
5.43.2.7	ds_str_concat	95
5.43.2.8	ds_str_concat_cstr	95
5.43.2.9	ds_str_create	96
5.43.2.10	ds_str_create_direct	96
5.43.2.11	ds_str_create_sprintf	96
5.43.2.12	ds_str_cstr	97
5.43.2.13	ds_str_decorate	97
5.43.2.14	ds_str_destroy	97
5.43.2.15	ds_str_destructor	97
5.43.2.16	ds_str_doubleval	97
5.43.2.17	ds_str_dup	98
5.43.2.18	ds_str_getline	98
5.43.2.19	ds_str_hash	98
5.43.2.20	ds_str_intval	98
5.43.2.21	ds_str_is_empty	99
5.43.2.22	ds_str_length	99
5.43.2.23	ds_str_size_to_fit	99
5.43.2.24	ds_str_split	99
5.43.2.25	ds_str_strchr	100
5.43.2.26	ds_str_substr_left	100
5.43.2.27	ds_str_substr_right	100

5.43.2.28 ds_str_trim	100
5.43.2.29 ds_str_trim_leading	101
5.43.2.30 ds_str_trim_trailing	101
5.43.2.31 ds_str_trunc	101
5.44 lib/datastruct/ds_str.h File Reference	101
5.44.1 Detailed Description	103
5.44.2 Typedef Documentation	103
5.44.2.1 ds_str	103
5.44.3 Function Documentation	104
5.44.3.1 ds_str_assign	104
5.44.3.2 ds_str_assign_cstr	104
5.44.3.3 ds_str_char_at_index	104
5.44.3.4 ds_str_clear	104
5.44.3.5 ds_str_compare	104
5.44.3.6 ds_str_compare_cstr	105
5.44.3.7 ds_str_concat	105
5.44.3.8 ds_str_concat_cstr	105
5.44.3.9 ds_str_create	105
5.44.3.10 ds_str_create_direct	106
5.44.3.11 ds_str_create_sprintf	106
5.44.3.12 ds_str_cstr	106
5.44.3.13 ds_str_decorate	107
5.44.3.14 ds_str_destroy	107
5.44.3.15 ds_str_destructor	107
5.44.3.16 ds_str_doubleval	107
5.44.3.17 ds_str_dup	107
5.44.3.18 ds_str_getline	108
5.44.3.19 ds_str_hash	108
5.44.3.20 ds_str_intval	108
5.44.3.21 ds_str_is_empty	109
5.44.3.22 ds_str_length	109
5.44.3.23 ds_str_size_to_fit	109
5.44.3.24 ds_str_split	109
5.44.3.25 ds_str_strchr	109
5.44.3.26 ds_str_substr_left	110
5.44.3.27 ds_str_substr_right	110
5.44.3.28 ds_str_trim	110
5.44.3.29 ds_str_trim_leading	110
5.44.3.30 ds_str_trim_trailing	111
5.44.3.31 ds_str_trunc	111

5.45 lib/datastruct/ds_vector.c File Reference	111
5.45.1 Detailed Description	112
5.45.2 Function Documentation	112
5.45.2.1 ds_vector_clear	112
5.45.2.2 ds_vector_create	113
5.45.2.3 ds_vector_destroy	113
5.45.2.4 ds_vector_destructor	113
5.45.2.5 ds_vector_element	113
5.45.2.6 ds_vector_get_next_data	113
5.45.2.7 ds_vector_seek_start	114
5.45.2.8 ds_vector_set	114
5.45.2.9 ds_vector_size	114
5.46 lib/datastruct/ds_vector.h File Reference	115
5.46.1 Detailed Description	116
5.46.2 Typedef Documentation	116
5.46.2.1 ds_vector	116
5.46.3 Function Documentation	116
5.46.3.1 ds_vector_clear	116
5.46.3.2 ds_vector_create	116
5.46.3.3 ds_vector_destroy	117
5.46.3.4 ds_vector_destructor	117
5.46.3.5 ds_vector_element	117
5.46.3.6 ds_vector_get_next_data	117
5.46.3.7 ds_vector_seek_start	118
5.46.3.8 ds_vector_set	118
5.46.3.9 ds_vector_size	118
5.47 lib/file_ops/config_file_read.c File Reference	118
5.47.1 Detailed Description	119
5.47.2 Macro Definition Documentation	119
5.47.2.1 CONFIG_MAP_SIZE	119
5.47.2.2 MAX_BUFFER_SIZE	120
5.47.3 Function Documentation	120
5.47.3.1 config_file_free	120
5.47.3.2 config_file_read	120
5.47.3.3 config_file_value	120
5.48 lib/file_ops/config_file_read.h File Reference	120
5.48.1 Detailed Description	122
5.48.2 Macro Definition Documentation	122
5.48.2.1 CONFIG_FILE_MALFORMED_FILE	122
5.48.2.2 CONFIG_FILE_NO_FILE	122

5.48.2.3	CONFIG_FILE_OK	122
5.48.3	Function Documentation	122
5.48.3.1	config_file_free	122
5.48.3.2	config_file_read	122
5.48.3.3	config_file_value	123
5.49	lib/file_ops/delim_file_read.c File Reference	123
5.49.1	Detailed Description	124
5.49.2	Macro Definition Documentation	124
5.49.2.1	MAX_LINE_SIZE	124
5.49.3	Function Documentation	124
5.49.3.1	delim_file_read	124
5.50	lib/file_ops/delim_file_read.h File Reference	124
5.50.1	Detailed Description	126
5.50.2	Function Documentation	126
5.50.2.1	delim_file_read	126
5.51	lib/file_ops/file_ops.h File Reference	126
5.51.1	Detailed Description	127
5.52	lib/gl_general/gl_errors.c File Reference	128
5.52.1	Detailed Description	128
5.52.2	Function Documentation	128
5.52.2.1	gl_error_quit	128
5.53	lib/gl_general/gl_errors.h File Reference	129
5.53.1	Detailed Description	129
5.53.2	Function Documentation	129
5.53.2.1	gl_error_quit	129
5.54	lib/gl_general/gl_general.h File Reference	129
5.54.1	Detailed Description	130
5.55	lib/gl_general/gl_logging.c File Reference	130
5.55.1	Detailed Description	131
5.55.2	Function Documentation	131
5.55.2.1	gl_log_msg	131
5.55.2.2	gl_set_logging	132
5.56	lib/gl_general/gl_logging.h File Reference	132
5.56.1	Detailed Description	132
5.56.2	Function Documentation	133
5.56.2.1	gl_log_msg	133
5.56.2.2	gl_set_logging	133
5.57	main.c File Reference	133
5.57.1	Detailed Description	134
5.57.2	Function Documentation	134

5.57.2.1	login	134
5.57.2.2	main	135
5.57.2.3	print_help_message	135
5.57.2.4	print_usage_message	135
5.57.2.5	print_version_message	135
5.57.2.6	test_functionality	135

Chapter 1

General Ledger.

General Ledger will be a fully-featured, multi-user, open-source general ledger system. The project is in the early stages of development.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

ds_list	9
ds_list_element	10
ds_map	11
ds_map_str	12
ds_record	13
ds_recordset	14
ds_str	15
ds_vector	15
kv_pair_node	16
params	17

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

config.c	Implementation of program configuration functionality	21
config.h	Interface to program configuration functionality	23
main.c	Main function for general_ledger	133
lib/database/database.h	User interface to database functionality	25
lib/database/db_connection.h	Interface to database connection functionality	27
lib/database/db_entities.c	Implementation of entities functionality	28
lib/database/db_entities.h	Interface to entities functionality	29
lib/database/db_internal.h	Internal library interface to database functionality	31
lib/database/db_query.h	Interface to database query functionality	32
lib/database/db_reporting.c	Implementation of database reporting functionality	33
lib/database/db_reporting.h	Interface to database reporting functionality	35
lib/database/db_sampledata.c	Implementation of database sample data functionality	36
lib/database/db_sampledata.h	Interface to database sample data functionality	37
lib/database/db_sql.h	Interface to database specific SQL strings	37
lib/database/db_structure.c	Implementation of database structure functionality	39
lib/database/db_structure.h	Interface to database structure functionality	41
lib/database/db_users.c	Implementation of users functionality	42
lib/database/db_users.h	Interface to users functionality	44
lib/database/dummy/db_dummy_create_entities_table_sql.c	Returns dummy SQL query to create entities table	45

lib/database/dummy/db_dummy_create_users_table_sql.c	Returns dummy SQL query to create users table	46
lib/database/dummy/db_dummy_drop_entities_table_sql.c	Returns dummy SQL query to drop entities table	46
lib/database/dummy/db_dummy_drop_users_table_sql.c	Returns dummy SQL query to drop users table	47
lib/database/dummy/db_dummy_general.c	Implementation of dummy database functionality	48
lib/database/dummy/db_dummy_list_entities_report_sql.c	Returns dummy SQL query to create list entities report	50
lib/database/dummy/db_dummy_list_users_report_sql.c	Returns dummy SQL query to create list users report	50
lib/database/mysql/db_mysql_create_entities_table_sql.c	Returns MYSQL SQL query to create entities table	51
lib/database/mysql/db_mysql_create_users_table_sql.c	Returns MYSQL SQL query to create users table	51
lib/database/mysql/db_mysql_drop_entities_table_sql.c	Returns MYSQL SQL query to drop entities table	52
lib/database/mysql/db_mysql_drop_users_table_sql.c	Returns MYSQL SQL query to drop users table	53
lib/database/mysql/db_mysql_general.c	Implementation of MYSQL database functionality	53
lib/database/mysql/db_mysql_list_entities_report_sql.c	Returns MYSQL SQL query to create list entities report	56
lib/database/mysql/db_mysql_list_users_report_sql.c	Returns MYSQL SQL query to create list users report	56
lib/datastruct/data_structures.h	Interface to data structures	57
lib/datastruct/ds_list.c	Implementation of generic doubly-linked list data structure	58
lib/datastruct/ds_list.h	Interface to generic doubly-linked list data structure	62
lib/datastruct/ds_map.c	Implementation of string-string hash map data structure	67
lib/datastruct/ds_map.h	Interface to string-string hash map data structure	69
lib/datastruct/ds_map_str.c	Implementation of string-string hash map data structure	72
lib/datastruct/ds_map_str.h	Interface to string-string hash map data structure	74
lib/datastruct/ds_record.c	Implementation of record database structure	76
lib/datastruct/ds_record.h	Interface to record data structure	80
lib/datastruct/ds_recordset.c	Implementation of query result set structure	84
lib/datastruct/ds_recordset.h	Interface to record set structure	88
lib/datastruct/ds_str.c	Implementation of string data structure	92
lib/datastruct/ds_str.h	Interface to string data structure	101
lib/datastruct/ds_vector.c	Implementation of generic doubly-linked vector data structure	111
lib/datastruct/ds_vector.h	Interface to generic doubly-linked vector data structure	115
lib/file_ops/config_file_read.c	Implementation of configuration file reading functionality	118

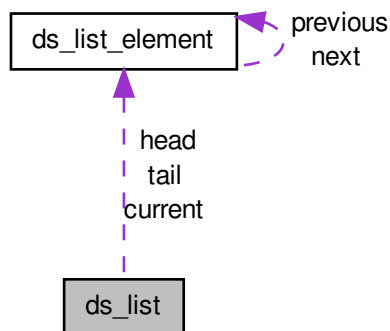
lib/file_ops/ config_file_read.h	
Interface to configuration file reading functionality	120
lib/file_ops/ delim_file_read.c	
Implementation of delimited file reading functionality	123
lib/file_ops/ delim_file_read.h	
Interface to delimited file reading functionality	124
lib/file_ops/ file_ops.h	
User interface to file operations functionality	126
lib/gl_general/ gl_errors.c	
Implementation of error functionality	128
lib/gl_general/ gl_errors.h	
Interface to error functionality	129
lib/gl_general/ gl_general.h	
User interface to logging and error functionality	129
lib/gl_general/ gl_logging.c	
Implementation of logging functionality	130
lib/gl_general/ gl_logging.h	
Interface to logging functionality	132

Chapter 4

Data Structure Documentation

4.1 ds_list Struct Reference

Collaboration diagram for ds_list:



Data Fields

- `size_t` `length`
- `bool` `free_on_delete`
- `struct ds_list_element *` `head`
- `struct ds_list_element *` `tail`
- `struct ds_list_element *` `current`
- `void(* data_destructor)(void *)`

4.1.1 Detailed Description

List data structure

4.1.2 Field Documentation

4.1.2.1 struct `ds_list_element*` `ds_list::current`

Pointer to current element

4.1.2.2 void(* `ds_list::data_destructor`)(void *)

Data destructor function

4.1.2.3 bool `ds_list::free_on_delete`

'Free on delete' flag

4.1.2.4 struct `ds_list_element*` `ds_list::head`

Pointer to head element

4.1.2.5 size_t `ds_list::length`

Length of list

4.1.2.6 struct `ds_list_element*` `ds_list::tail`

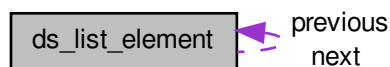
Pointer to tail element

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_list.c](#)

4.2 ds_list_element Struct Reference

Collaboration diagram for `ds_list_element`:



Data Fields

- void * [data](#)
- struct `ds_list_element` * [previous](#)
- struct `ds_list_element` * [next](#)

4.2.1 Detailed Description

List element data structure

4.2.2 Field Documentation

4.2.2.1 void* ds_list_element::data

Pointer to data

4.2.2.2 struct ds_list_element* ds_list_element::next

Pointer to next element

4.2.2.3 struct ds_list_element* ds_list_element::previous

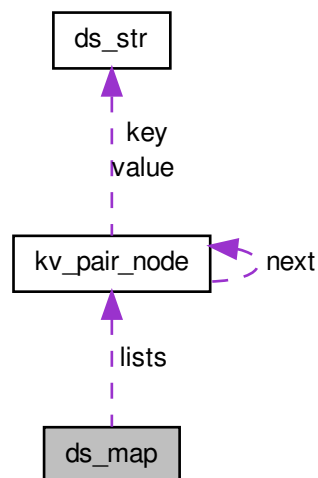
Pointer to previous element

The documentation for this struct was generated from the following file:

- lib/datastruct/[ds_list.c](#)

4.3 ds_map Struct Reference

Collaboration diagram for ds_map:



Data Fields

- struct [kv_pair_node](#) ** `lists`
- size_t [hash_size](#)

4.3.1 Detailed Description

Structure to hold a hash map

4.3.2 Field Documentation

4.3.2.1 `size_t ds_map::hash_size`

Size of array of lists

4.3.2.2 `struct kv_pair_node** ds_map::lists`

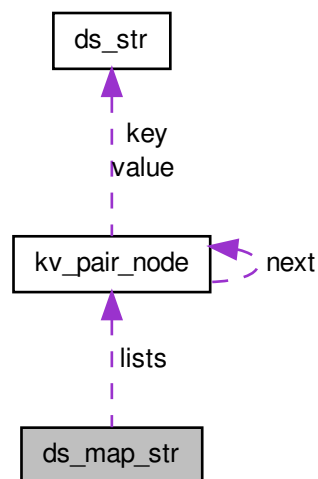
Pointer to array of lists

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_map.c](#)

4.4 `ds_map_str` Struct Reference

Collaboration diagram for `ds_map_str`:



Data Fields

- struct [kv_pair_node](#) ** `lists`
- `size_t` [hash_size](#)

4.4.1 Detailed Description

Structure to hold a hash map

4.4.2 Field Documentation

4.4.2.1 `size_t ds_map_str::hash_size`

Size of array of lists

4.4.2.2 `struct kv_pair_node** ds_map_str::lists`

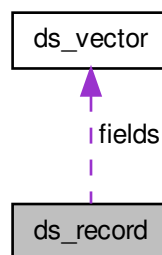
Pointer to array of lists

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_map_str.c](#)

4.5 ds_record Struct Reference

Collaboration diagram for ds_record:



Data Fields

- struct [ds_vector](#) * `fields`

4.5.1 Detailed Description

Vector data structure

4.5.2 Field Documentation

4.5.2.1 `struct ds_vector* ds_record::fields`

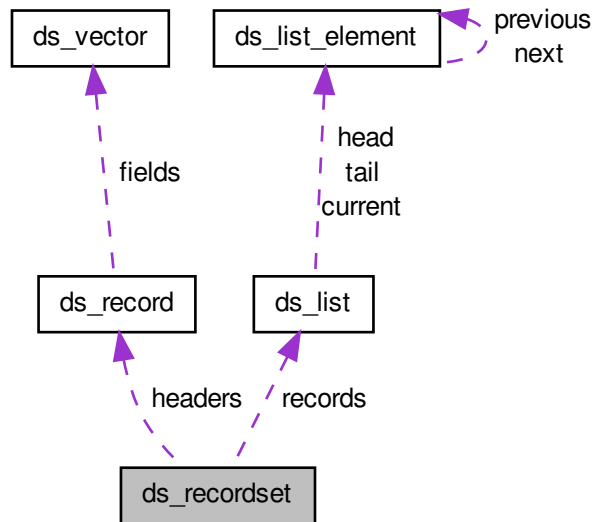
Vector of fields

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_record.c](#)

4.6 ds_recordset Struct Reference

Collaboration diagram for ds_recordset:



Data Fields

- `size_t num_fields`
- `size_t * field_lengths`
- `ds_record headers`
- `ds_list records`

4.6.1 Detailed Description

Result set structure

4.6.2 Field Documentation

4.6.2.1 `size_t * ds_recordset::field_lengths`

Lengths of the longest fields

4.6.2.2 `ds_record ds_recordset::headers`

A list of field headers

4.6.2.3 `size_t ds_recordset::num_fields`

The number of fields in a record

4.6.2.4 ds_list ds_recordset::records

A list of records

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_recordset.c](#)

4.7 ds_str Struct Reference

Data Fields

- `char *` [data](#)
- `size_t` [length](#)
- `size_t` [capacity](#)

4.7.1 Detailed Description

Structure to contain string

4.7.2 Field Documentation

4.7.2.1 size_t ds_str::capacity

The size of the `data` buffer

4.7.2.2 char* ds_str::data

The data in C-style string format

4.7.2.3 size_t ds_str::length

The length of the string

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_str.c](#)

4.8 ds_vector Struct Reference

Data Fields

- `size_t` [size](#)
- `size_t` [current](#)
- `bool` [free_on_delete](#)
- `void **` [data](#)
- `void(*` [data_destructor](#) `)(void *)`

4.8.1 Detailed Description

Vector data structure

4.8.2 Field Documentation

4.8.2.1 `size_t ds_vector::current`

Current position

4.8.2.2 `void** ds_vector::data`

Data array

4.8.2.3 `void(* ds_vector::data_destructor)(void *)`

Data destructor function

4.8.2.4 `bool ds_vector::free_on_delete`

'Free on delete' flag

4.8.2.5 `size_t ds_vector::size`

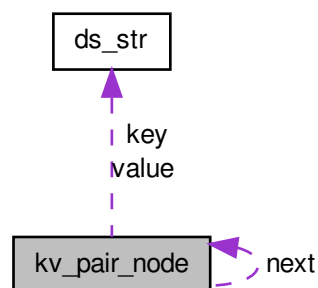
Size of vector

The documentation for this struct was generated from the following file:

- [lib/datastruct/ds_vector.c](#)

4.9 kv_pair_node Struct Reference

Collaboration diagram for kv_pair_node:



Data Fields

- `char * key`
- `char * value`
- `struct kv_pair_node * next`

- [ds_str key](#)
- [ds_str value](#)

4.9.1 Detailed Description

Structure to hold a key-value pair node

4.9.2 Field Documentation

4.9.2.1 `ds_str kv_pair_node::key`

A pointer to the key

4.9.2.2 `char* kv_pair_node::key`

A pointer to the key

4.9.2.3 `struct kv_pair_node * kv_pair_node::next`

A pointer to the next node

4.9.2.4 `ds_str kv_pair_node::value`

A pointer to the value

4.9.2.5 `char* kv_pair_node::value`

A pointer to the value

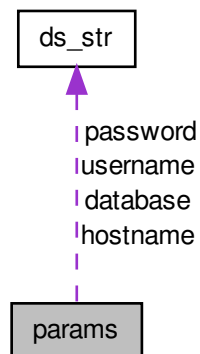
The documentation for this struct was generated from the following files:

- [lib/datastruct/ds_map.c](#)
- [lib/datastruct/ds_map_str.c](#)

4.10 params Struct Reference

```
#include <config.h>
```

Collaboration diagram for params:



Data Fields

- [ds_str hostname](#)
- [ds_str database](#)
- [ds_str username](#)
- [ds_str password](#)
- [bool help](#)
- [bool version](#)
- [bool create](#)
- [bool delete_data](#)
- [bool sample](#)
- [bool list_users](#)
- [bool list_entities](#)

4.10.1 Detailed Description

Structure to hold program parameters

4.10.2 Field Documentation

4.10.2.1 `bool params::create`

Create structure option set

4.10.2.2 `ds_str params::database`

Database name

4.10.2.3 `bool params::delete_data`

Delete structure option set

4.10.2.4 bool params::help

Help option set

4.10.2.5 ds_str params::hostname

Database hostname

4.10.2.6 bool params::list_entities

List entities option set

4.10.2.7 bool params::list_users

List users option set

4.10.2.8 ds_str params::password

Password for database access

4.10.2.9 bool params::sample

Load sample data option set

4.10.2.10 ds_str params::username

Username for database access

4.10.2.11 bool params::version

Version option set

The documentation for this struct was generated from the following file:

- [config.h](#)

Chapter 5

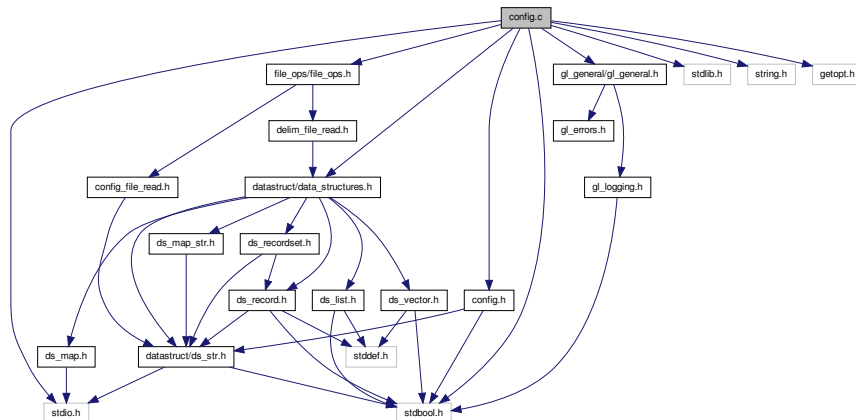
File Documentation

5.1 config.c File Reference

Implementation of program configuration functionality.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include "config.h"
#include "file_ops/file_ops.h"
#include "datastruct/data_structures.h"
#include "gl_general/gl_general.h"
```

Include dependency graph for config.c:



Macros

- `#define _XOPEN_SOURCE 500`

Functions

- `struct params * params_init (void)`
Initializes a parameters structure.

- void `params_free` (struct `params` *`params`)
Frees a parameter structure.
- bool `get_configuration` (struct `params` *`params`)
Gets parameters from a configuration file.
- bool `get_cmdline_options` (int `argc`, char **`argv`, struct `params` *`params`)
Gets parameters from the command line.

5.1.1 Detailed Description

Implementation of program configuration functionality. Gets program configuration options from the command line and/or a configuration file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.1.2 Macro Definition Documentation

5.1.2.1 `#define XOPEN.SOURCE 500`

UNIX feature test macro

5.1.3 Function Documentation

5.1.3.1 bool `get_cmdline_options` (int *argc*, char ** *argv*, struct `params` * *params*)

Gets parameters from the command line.

Parameters

<i>argc</i>	<i>argc</i> as passed to <code>main()</code> .
<i>argv</i>	<i>argv</i> as passed to <code>main()</code> .
<i>params</i>	A pointer to a parameters structure to populate.

Returns

`false` if an unrecognized command line option was specified, `true` otherwise.

5.1.3.2 bool `get_configuration` (struct `params` * *params*)

Gets parameters from a configuration file.

Parameters

<i>params</i>	A pointer to a parameters structure to populate.
---------------	--

Returns

true on success, false otherwise.

5.1.3.3 void params_free (struct params * params)

Frees a parameter structure.

Parameters

<i>params</i>	A pointer to the structure to free.
---------------	-------------------------------------

5.1.3.4 struct params* params_init (void) [read]

Initializes a parameters structure.

Returns

An initialized parameters structure.

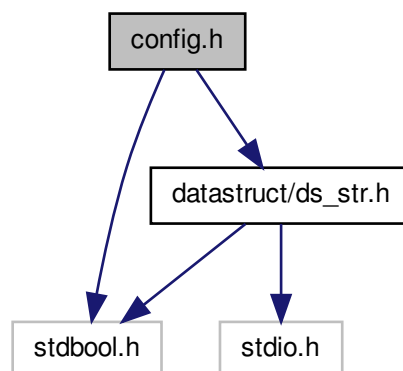
5.2 config.h File Reference

Interface to program configuration functionality.

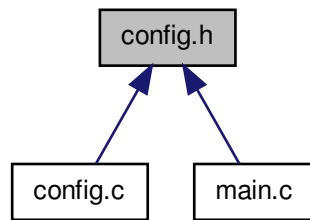
```
#include <stdbool.h>
```

```
#include "datastruct/ds_str.h"
```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [params](#)

Functions

- struct [params](#) * [params_init](#) (void)
Initializes a parameters structure.
- void [params_free](#) (struct [params](#) *[params](#))
Frees a parameter structure.
- bool [get_configuration](#) (struct [params](#) *[params](#))
Gets parameters from a configuration file.
- bool [get_cmdline_options](#) (int argc, char **argv, struct [params](#) *[params](#))
Gets parameters from the command line.

5.2.1 Detailed Description

Interface to program configuration functionality. Gets program configuration options from the command line and/or a configuration file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.2.2 Function Documentation

5.2.2.1 bool [get_cmdline_options](#) (int *argc*, char ** *argv*, struct [params](#) * *params*)

Gets parameters from the command line.

Parameters

<i>argc</i>	<code>argc</code> as passed to <code>main()</code> .
<i>argv</i>	<code>argv</code> as passed to <code>main()</code> .
<i>params</i>	A pointer to a parameters structure to populate.

Returns

`false` if an unrecognized command line option was specified, `true` otherwise.

5.2.2.2 bool get_configuration (struct params * params)

Gets parameters from a configuration file.

Parameters

<i>params</i>	A pointer to a parameters structure to populate.
---------------	--

Returns

`true` on success, `false` otherwise.

5.2.2.3 void params_free (struct params * params)

Frees a parameter structure.

Parameters

<i>params</i>	A pointer to the structure to free.
---------------	-------------------------------------

5.2.2.4 struct params* params_init (void) [read]

Initializes a parameters structure.

Returns

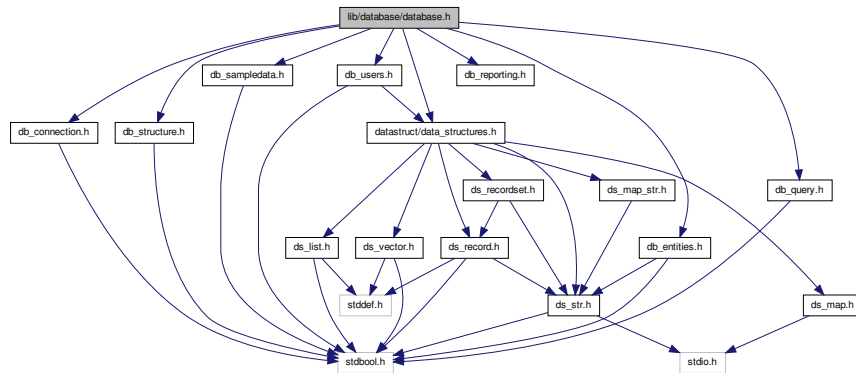
An initialized parameters structure.

5.3 lib/database/database.h File Reference

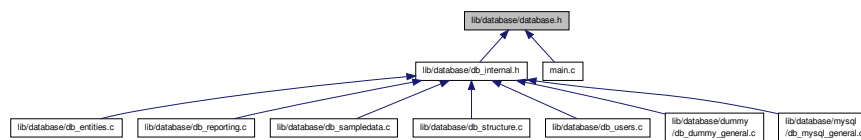
User interface to database functionality.

```
#include "datastruct/data_structures.h"
#include "db_connection.h"
#include "db_structure.h"
#include "db_query.h"
#include "db_sampledata.h"
#include "db_reporting.h"
#include "db_users.h"
#include "db_entities.h"
```

Include dependency graph for database.h:



This graph shows which files directly or indirectly include this file:



5.3.1 Detailed Description

User interface to database functionality.

Author

Paul Griffiths

Copyright

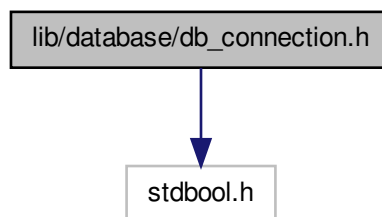
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.4 lib/database/db_connection.h File Reference

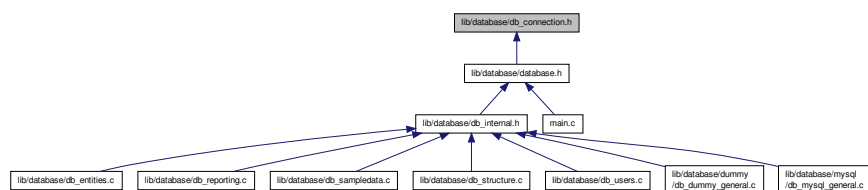
Interface to database connection functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_connection.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool **db_connect** (const char *host, const char *database, const char *username, const char *password)
Connects to a database.
- void **db_close** (void)
Disconnects from a database.

5.4.1 Detailed Description

Interface to database connection functionality. Function implementations are provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.4.2 Function Documentation

5.4.2.1 `bool db_connect (const char * host, const char * database, const char * username, const char * password)`

Connects to a database.

Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

Returns

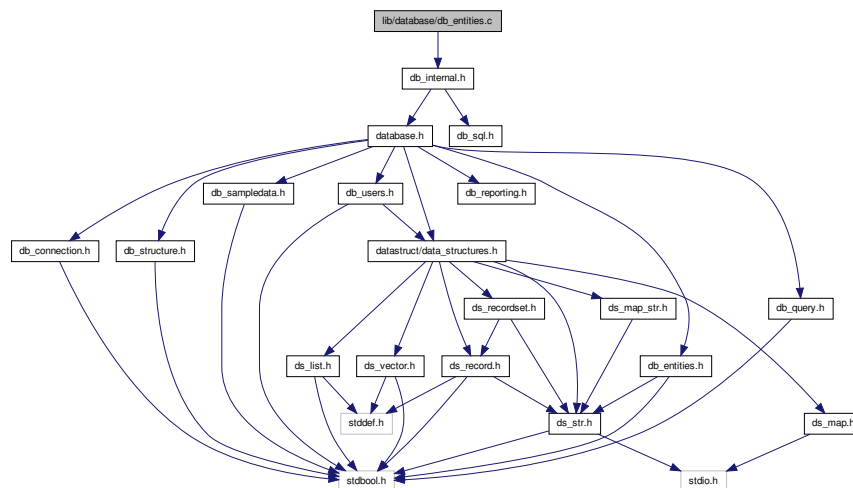
`true` if the connection was successfully made, `false` otherwise.

5.5 lib/database/db_entities.c File Reference

Implementation of entities functionality.

```
#include "db_internal.h"
```

Include dependency graph for `db_entities.c`:



Functions

- `bool db_create_entities_table (void)`
Creates the entities table in the database.
- `bool db_drop_entities_table (void)`
Drops the entities table in the database.
- `ds_str db_list_entities_report (void)`
Creates a report listing all entities.

5.5.1 Detailed Description

Implementation of entities functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.5.2 Function Documentation

5.5.2.1 `bool db_create_entities_table (void)`

Creates the entities table in the database.

Returns

`true` on success, `false` on failure.

5.5.2.2 `bool db_drop_entities_table (void)`

Drops the entities table in the database.

Returns

`true` on success, `false` on failure.

5.5.2.3 `ds_str db_list_entities_report (void)`

Creates a report listing all entities.

Returns

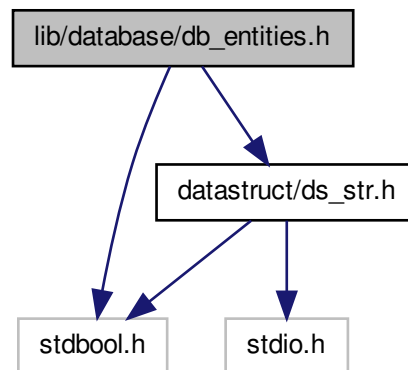
A `ds_str` containing the report.

5.6 lib/database/db_entities.h File Reference

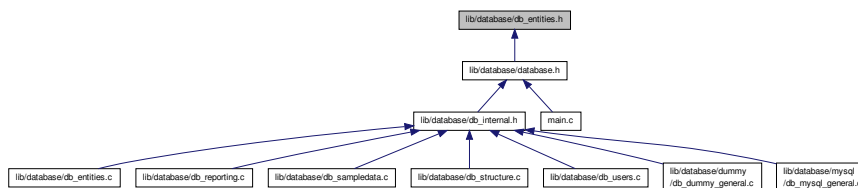
Interface to entities functionality.

```
#include <stdbool.h>
#include "datastruct/ds_str.h"
```

Include dependency graph for `db_entities.h`:



This graph shows which files directly or indirectly include this file:



Functions

- `bool db_create_entities_table (void)`
Creates the entities table in the database.
- `bool db_drop_entities_table (void)`
Drops the entities table in the database.
- `ds_str db_list_entities_report (void)`
Creates a report listing all entities.

5.6.1 Detailed Description

Interface to entities functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.6.2 Function Documentation

5.6.2.1 bool db.create_entities_table (void)

Creates the entities table in the database.

Returns

true on success, false on failure.

5.6.2.2 bool db.drop_entities_table (void)

Drops the entities table in the database.

Returns

true on success, false on failure.

5.6.2.3 ds_str db.list_entities_report (void)

Creates a report listing all entities.

Returns

A [ds_str](#) containing the report.

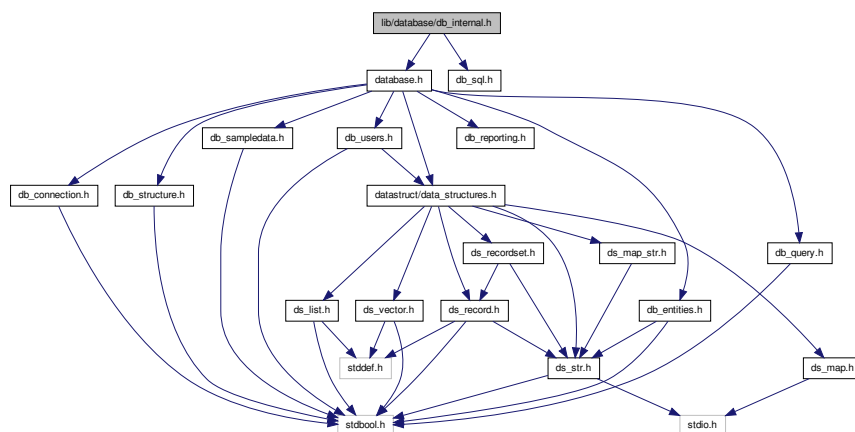
5.7 lib/database/db_internal.h File Reference

Internal library interface to database functionality.

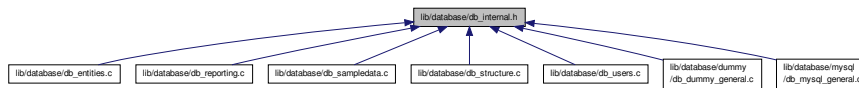
```
#include "database.h"
```

```
#include "db_sql.h"
```

Include dependency graph for db_internal.h:



This graph shows which files directly or indirectly include this file:



5.7.1 Detailed Description

Internal library interface to database functionality. The library interface includes the individual SQL functions which should be encapsulated from the user.

Author

Paul Griffiths

Copyright

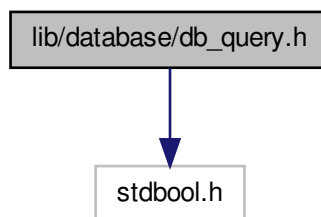
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.8 lib/database/db_query.h File Reference

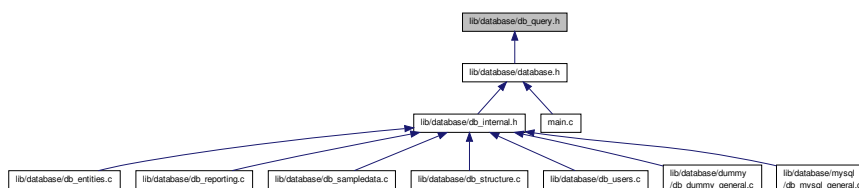
Interface to database query functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_query.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool `db_execute_query` (const char *query)
Executes an SQL query on the database.

5.8.1 Detailed Description

Interface to database query functionality. Function implementations are provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.8.2 Function Documentation

5.8.2.1 bool db_execute_query (const char * query)

Executes an SQL query on the database.

Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

Returns

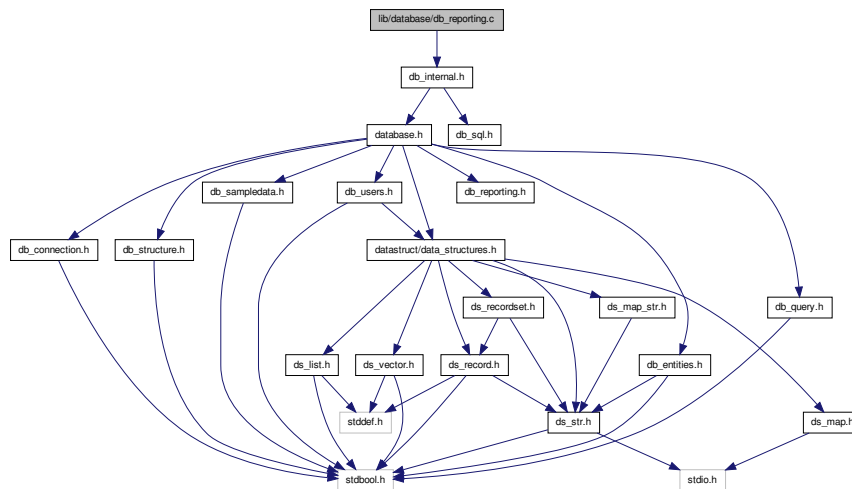
`true` if the query was successfully executed, `false` otherwise.

5.9 lib/database/db_reporting.c File Reference

Implementation of database reporting functionality.

```
#include "db_internal.h"
```

Include dependency graph for db_reporting.c:



Functions

- [ds_str db_create_report_from_query](#) (const char *query)
Creates a text report from a query.

5.9.1 Detailed Description

Implementation of database reporting functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.9.2 Function Documentation

5.9.2.1 ds_str db_create_report_from_query (const char * query)

Creates a text report from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

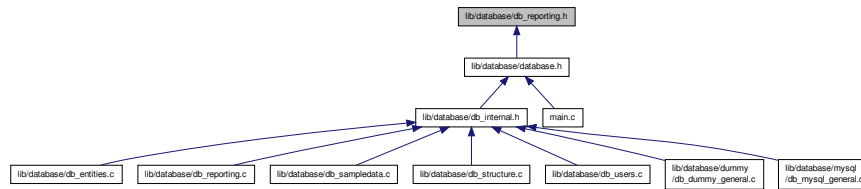
Returns

A [ds_str](#) containing the report, or NULL on failure.

5.10 lib/database/db_reporting.h File Reference

Interface to database reporting functionality.

This graph shows which files directly or indirectly include this file:



Functions

- [ds_str db_create_report_from_query](#) (const char *query)
Creates a text report from a query.
- [ds_recordset db_create_recordset_from_query](#) (const char *query)
Creates a [ds_recordset](#) from a query.

5.10.1 Detailed Description

Interface to database reporting functionality. Function implementations may be provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.10.2 Function Documentation

5.10.2.1 ds_recordset db_create_recordset_from_query (const char * query)

Creates a [ds_recordset](#) from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

Returns

A [ds_recordset](#) containing the query result, or NULL on failure.

5.10.2.2 ds_str db_create_report_from_query (const char * query)

Creates a text report from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

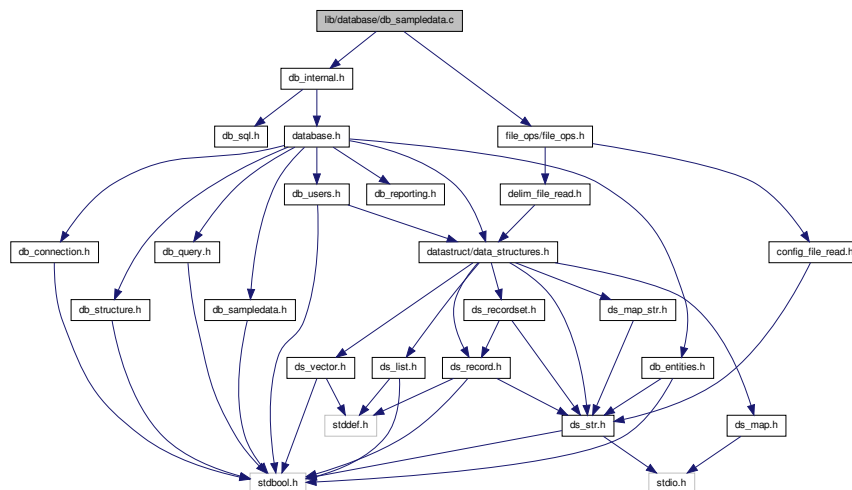
Returns

A [ds_str](#) containing the report, or NULL on failure.

5.11 lib/database/db_sampledata.c File Reference

Implementation of database sample data functionality.

```
#include "db_internal.h"
#include "file_ops/file_ops.h"
Include dependency graph for db_sampledata.c:
```



Functions

- bool [db_load_sample_data](#) (void)
Loads sample data into the database.

5.11.1 Detailed Description

Implementation of database sample data functionality.

Author

Paul Griffiths

Copyright

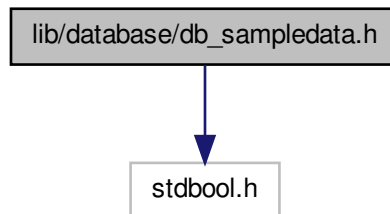
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.12 lib/database/db_sampledata.h File Reference

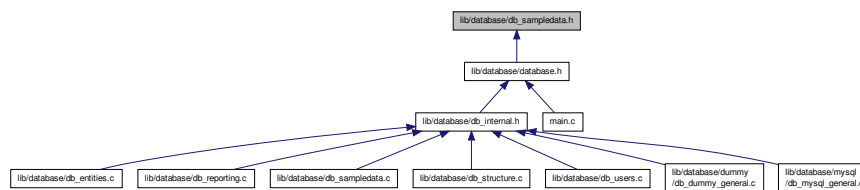
Interface to database sample data functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_sampledata.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool [db_load_sample_data](#) (void)
Loads sample data into the database.

5.12.1 Detailed Description

Interface to database sample data functionality.

Author

Paul Griffiths

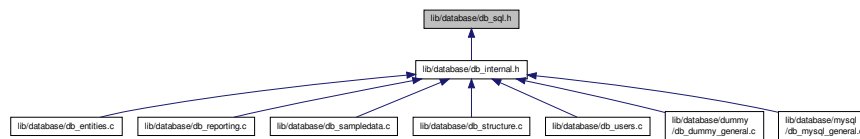
Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.13 lib/database/db_sql.h File Reference

Interface to database specific SQL strings.

This graph shows which files directly or indirectly include this file:



Functions

- `const char * db_create_users_table_sql (void)`
Returns the SQL query to create the users table.
- `const char * db_drop_users_table_sql (void)`
Returns the SQL query to drop the users table.
- `const char * db_list_users_report_sql (void)`
Returns the SQL query to run the "list users" report.
- `const char * db_create_entities_table_sql (void)`
Returns the SQL query to create the entities table.
- `const char * db_drop_entities_table_sql (void)`
Returns the SQL query to drop the entities table.
- `const char * db_list_entities_report_sql (void)`
Returns the SQL query to run the "list entities" report.

5.13.1 Detailed Description

Interface to database specific SQL strings. Function implementations are provided by the individual database components.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.13.2 Function Documentation

5.13.2.1 `const char* db_create_entities_table_sql (void)`

Returns the SQL query to create the entities table.

Returns

The SQL query.

5.13.2.2 `const char* db_create_users_table_sql (void)`

Returns the SQL query to create the users table.

Returns

The SQL query.

5.13.2.3 `const char* db_drop_entities_table_sql (void)`

Returns the SQL query to drop the entities table.

Returns

The SQL query.

5.13.2.4 `const char* db_drop_users_table_sql (void)`

Returns the SQL query to drop the users table.

Returns

The SQL query.

5.13.2.5 `const char* db_list_entities_report_sql (void)`

Returns the SQL query to run the "list entities" report.

Returns

The SQL query.

5.13.2.6 `const char* db_list_users_report_sql (void)`

Returns the SQL query to run the "list users" report.

Returns

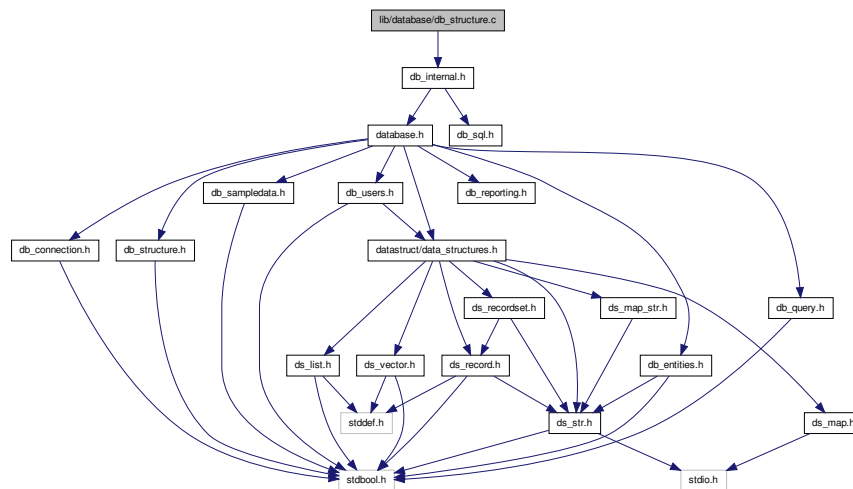
The SQL query.

5.14 lib/database/db_structure.c File Reference

Implementation of database structure functionality.

```
#include "db_internal.h"
```

Include dependency graph for db_structure.c:



Functions

- bool [db_create_database_structure](#) (void)
Creates an empty database structure.
- bool [db_delete_database_structure](#) (void)
Deletes the database structure.

5.14.1 Detailed Description

Implementation of database structure functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.14.2 Function Documentation

5.14.2.1 bool db_create_database_structure (void)

Creates an empty database structure.

Returns

true on success, false on failure.

5.14.2.2 bool db_delete_database_structure (void)

Deletes the database structure.

Returns

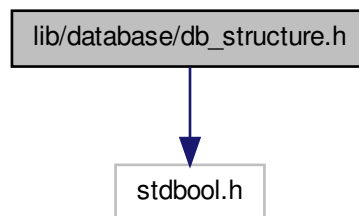
true on success, false on failure.

5.15 lib/database/db_structure.h File Reference

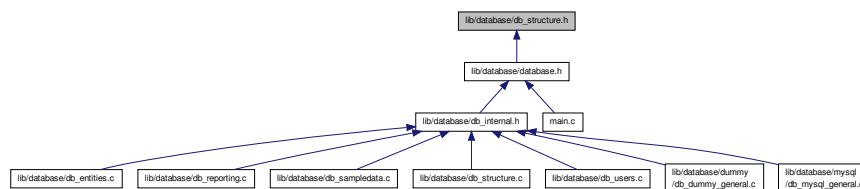
Interface to database structure functionality.

```
#include <stdbool.h>
```

Include dependency graph for db_structure.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool [db_create_database_structure](#) (void)
Creates an empty database structure.
- bool [db_delete_database_structure](#) (void)
Deletes the database structure.

5.15.1 Detailed Description

Interface to database structure functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.15.2 Function Documentation**5.15.2.1 bool db_create_database_structure (void)**

Creates an empty database structure.

Returns

true on success, false on failure.

5.15.2.2 bool db_delete_database_structure (void)

Deletes the database structure.

Returns

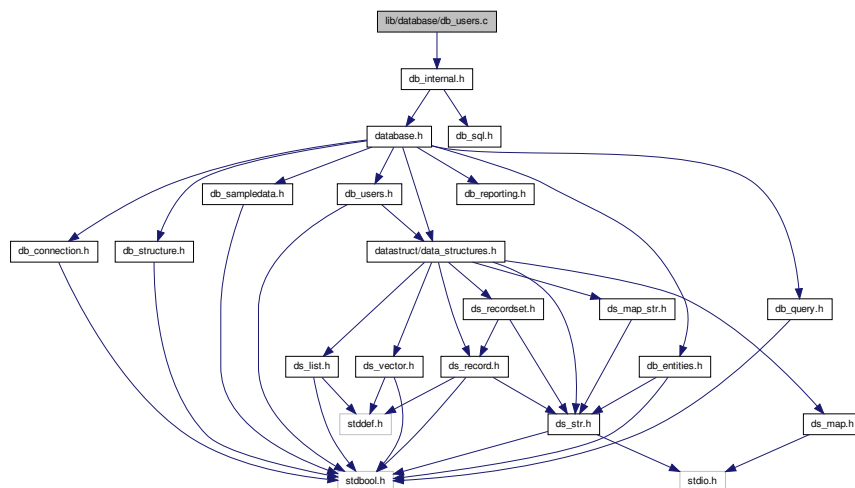
true on success, false on failure.

5.16 lib/database/db_users.c File Reference

Implementation of users functionality.

```
#include "db_internal.h"
```

Include dependency graph for db_users.c:



Functions

- `bool db_create_users_table (void)`
Creates the users table in the database.
- `bool db_drop_users_table (void)`
Drops the users table from the database.
- `ds_str db_list_users_report (void)`
Creates a report listing all users.

5.16.1 Detailed Description

Implementation of users functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.16.2 Function Documentation

5.16.2.1 `bool db_create_users_table (void)`

Creates the users table in the database.

Returns

`true` on success, `false` on failure.

5.16.2.2 `bool db_drop_users_table (void)`

Drops the users table from the database.

Returns

`true` on success, `false` on failure.

5.16.2.3 `ds_str db_list_users_report (void)`

Creates a report listing all users.

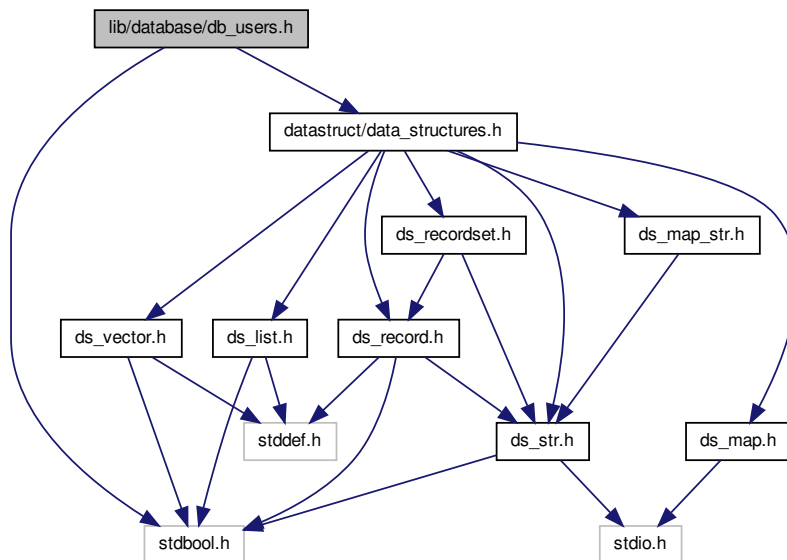
Returns

A `ds_str` containing the report.

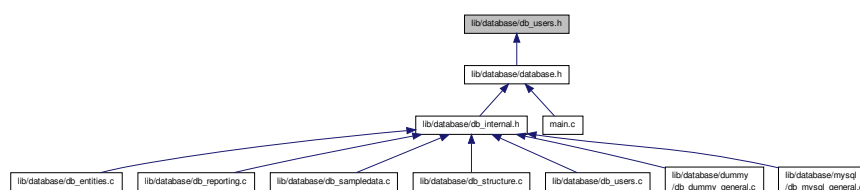
5.17 lib/database/db_users.h File Reference

Interface to users functionality.

```
#include <stdbool.h>
#include "datastruct/data_structures.h"
Include dependency graph for db_users.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- bool [db_create_users_table](#) (void)
Creates the users table in the database.
- bool [db_drop_users_table](#) (void)
Drops the users table from the database.
- [ds_str db_list_users_report](#) (void)
Creates a report listing all users.

5.17.1 Detailed Description

Interface to users functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.17.2 Function Documentation**5.17.2.1 bool db_create_users_table (void)**

Creates the users table in the database.

Returns

`true` on success, `false` on failure.

5.17.2.2 bool db_drop_users_table (void)

Drops the users table from the database.

Returns

`true` on success, `false` on failure.

5.17.2.3 ds_str db_list_users_report (void)

Creates a report listing all users.

Returns

A `ds_str` containing the report.

5.18 lib/database/dummy/db_dummy_create_entities_table_sql.c File Reference

Returns dummy SQL query to create entities table.

Functions

- `const char * db_create_entities_table_sql (void)`
Returns the SQL query to create the entities table.

5.18.1 Detailed Description

Returns dummy SQL query to create entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.18.2 Function Documentation

5.18.2.1 `const char* db_create_entities_table_sql (void)`

Returns the SQL query to create the entities table.

Returns

The SQL query.

5.19 `lib/database/dummy/db_dummy_create_users_table_sql.c` File Reference

Returns dummy SQL query to create users table.

Functions

- `const char * db_create_users_table_sql (void)`
Returns the SQL query to create the users table.

5.19.1 Detailed Description

Returns dummy SQL query to create users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.19.2 Function Documentation

5.19.2.1 `const char* db_create_users_table_sql (void)`

Returns the SQL query to create the users table.

Returns

The SQL query.

5.20 `lib/database/dummy/db_dummy_drop_entities_table_sql.c` File Reference

Returns dummy SQL query to drop entities table.

Functions

- `const char * db_drop_entities_table_sql (void)`
Returns the SQL query to drop the entities table.

5.20.1 Detailed Description

Returns dummy SQL query to drop entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.20.2 Function Documentation

5.20.2.1 `const char* db_drop_entities_table_sql (void)`

Returns the SQL query to drop the entities table.

Returns

The SQL query.

5.21 lib/database/dummy/db_dummy_drop_users_table_sql.c File Reference

Returns dummy SQL query to drop users table.

Functions

- `const char * db_drop_users_table_sql (void)`
Returns the SQL query to drop the users table.

5.21.1 Detailed Description

Returns dummy SQL query to drop users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.21.2 Function Documentation

5.21.2.1 `const char* db_drop_users_table_sql (void)`

Returns the SQL query to drop the users table.

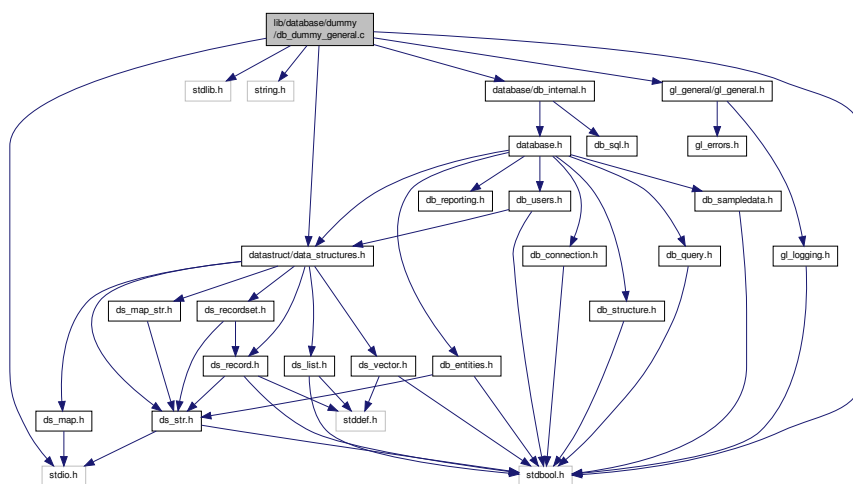
Returns

The SQL query.

5.22 lib/database/dummy/db_dummy_general.c File Reference

Implementation of dummy database functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "gl_general/gl_general.h"
#include "database/db_internal.h"
#include "datastruct/data_structures.h"
Include dependency graph for db_dummy_general.c:
```



Macros

- `#define _XOPEN_SOURCE 600`

Functions

- `bool db_connect (const char *host, const char *database, const char *username, const char *password)`
Connects to a database.
- `void db_close (void)`
Disconnects from a database.
- `bool db_execute_query (const char *query)`
Executes an SQL query on the database.
- `ds_recordset db_create_recordset_from_query (const char *query)`
Creates a [ds_recordset](#) from a query.

5.22.1 Detailed Description

Implementation of dummy database functionality. This module is useful when compiling for testing purpose on a system without any of the supported database development libraries available.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.22.2 Macro Definition Documentation

5.22.2.1 #define _XOPEN_SOURCE 600

UNIX feature test macro

5.22.3 Function Documentation

5.22.3.1 bool db_connect (const char * *host*, const char * *database*, const char * *username*, const char * *password*)

Connects to a database.

Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

Returns

`true` if the connection was successfully made, `false` otherwise.

5.22.3.2 ds_recordset db_create_recordset_from_query (const char * *query*)

Creates a [ds_recordset](#) from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

Returns

A [ds_recordset](#) containing the query result, or `NULL` on failure.

5.22.3.3 bool db_execute_query (const char * *query*)

Executes an SQL query on the database.

Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

Returns

`true` if the query was successfully executed, `false` otherwise.

5.23 lib/database/dummy/db_dummy_list_entities_report_sql.c File Reference

Returns dummy SQL query to create list entities report.

Functions

- `const char * db_list_entities_report_sql (void)`
Returns the SQL query to run the "list entities" report.

5.23.1 Detailed Description

Returns dummy SQL query to create list entities report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.23.2 Function Documentation

5.23.2.1 `const char* db_list_entities_report_sql (void)`

Returns the SQL query to run the "list entities" report.

Returns

The SQL query.

5.24 lib/database/dummy/db_dummy_list_users_report_sql.c File Reference

Returns dummy SQL query to create list users report.

Functions

- `const char * db_list_users_report_sql (void)`
Returns the SQL query to run the "list users" report.

5.24.1 Detailed Description

Returns dummy SQL query to create list users report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.24.2 Function Documentation

5.24.2.1 `const char* db_list_users_report_sql (void)`

Returns the SQL query to run the "list users" report.

Returns

The SQL query.

5.25 lib/database/mysql/db_mysql_create_entities_table_sql.c File Reference

Returns MYSQL SQL query to create entities table.

Functions

- `const char * db_create_entities_table_sql (void)`
Returns the SQL query to create the entities table.

5.25.1 Detailed Description

Returns MYSQL SQL query to create entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.25.2 Function Documentation

5.25.2.1 `const char* db_create_entities_table_sql (void)`

Returns the SQL query to create the entities table.

Returns

The SQL query.

5.26 lib/database/mysql/db_mysql_create_users_table_sql.c File Reference

Returns MYSQL SQL query to create users table.

Functions

- `const char * db_create_users_table_sql (void)`
Returns the SQL query to create the users table.

5.26.1 Detailed Description

Returns MYSQL SQL query to create users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.26.2 Function Documentation

5.26.2.1 `const char* db_create_users_table_sql (void)`

Returns the SQL query to create the users table.

Returns

The SQL query.

5.27 lib/database/mysql/db_mysql_drop_entities_table_sql.c File Reference

Returns MYSQL SQL query to drop entities table.

Functions

- `const char * db_drop_entities_table_sql (void)`
Returns the SQL query to drop the entities table.

5.27.1 Detailed Description

Returns MYSQL SQL query to drop entities table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.27.2 Function Documentation

5.27.2.1 const char* db_drop_entities_table_sql (void)

Returns the SQL query to drop the entities table.

Returns

The SQL query.

5.28 lib/database/mysql/db_mysql_drop_users_table_sql.c File Reference

Returns MYSQL SQL query to drop users table.

Functions

- const char * [db_drop_users_table_sql](#) (void)
Returns the SQL query to drop the users table.

5.28.1 Detailed Description

Returns MYSQL SQL query to drop users table.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.28.2 Function Documentation

5.28.2.1 const char* db_drop_users_table_sql (void)

Returns the SQL query to drop the users table.

Returns

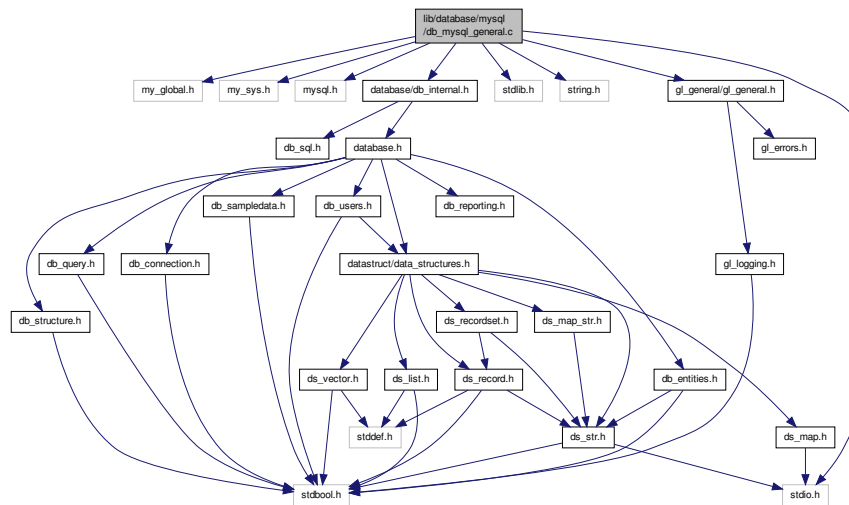
The SQL query.

5.29 lib/database/mysql/db_mysql_general.c File Reference

Implementation of MYSQL database functionality.

```
#include <my_global.h>
#include <my_sys.h>
#include <mysql.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "database/db_internal.h"
```

Include dependency graph for `db_mysql_general.c`:



Functions

- bool `db_connect` (const char *host, const char *database, const char *username, const char *password)
Connects to a database.
- void `db_close` (void)
Disconnects from a database.
- bool `db_execute_query` (const char *query)
Executes an SQL query on the database.
- `ds_recordset db_create_recordset_from_query` (const char *query)
Creates a `ds_recordset` from a query.

Variables

- MYSQL * `main_mss` = NULL
- MYSQL * `conn_mss` = NULL

5.29.1 Detailed Description

Implementation of MYSQL database functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.29.2 Function Documentation

5.29.2.1 `bool db_connect (const char * host, const char * database, const char * username, const char * password)`

Connects to a database.

Parameters

<i>host</i>	The hostname.
<i>database</i>	The database name.
<i>username</i>	The username with which to connect.
<i>password</i>	The password for the specified user.

Returns

`true` if the connection was successfully made, `false` otherwise.

5.29.2.2 `ds_recordset db_create_recordset_from_query (const char * query)`

Creates a [ds_recordset](#) from a query.

Parameters

<i>query</i>	The SELECT query to run.
--------------	--------------------------

Returns

A [ds_recordset](#) containing the query result, or `NULL` on failure.

5.29.2.3 `bool db_execute_query (const char * query)`

Executes an SQL query on the database.

Parameters

<i>query</i>	The query to execute.
--------------	-----------------------

Returns

`true` if the query was successfully executed, `false` otherwise.

5.29.3 Variable Documentation

5.29.3.1 `MYSQL* conn_mss = NULL`

MYSQL connection object.

5.29.3.2 `MYSQL* main_mss = NULL`

MYSQL initialization object.

5.30 lib/database/mysql/db_mysql_list_entities_report_sql.c File Reference

Returns MYSQL SQL query to create list entities report.

Functions

- const char * [db_list_entities_report_sql](#) (void)
Returns the SQL query to run the "list entities" report.

5.30.1 Detailed Description

Returns MYSQL SQL query to create list entities report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.30.2 Function Documentation

5.30.2.1 const char* db_list_entities_report_sql (void)

Returns the SQL query to run the "list entities" report.

Returns

The SQL query.

5.31 lib/database/mysql/db_mysql_list_users_report_sql.c File Reference

Returns MYSQL SQL query to create list users report.

Functions

- const char * [db_list_users_report_sql](#) (void)
Returns the SQL query to run the "list users" report.

5.31.1 Detailed Description

Returns MYSQL SQL query to create list users report.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.32.1 Detailed Description

Interface to data structures.

Author

Paul Griffiths

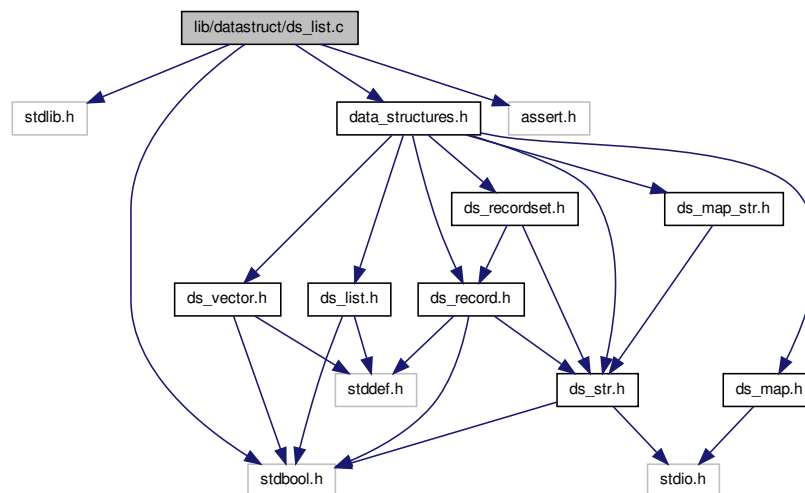
Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.33 lib/datastruct/ds_list.c File Reference

Implementation of generic doubly-linked list data structure.

```
#include <stdlib.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_list.c:
```



Data Structures

- struct [ds_list_element](#)
- struct [ds_list](#)

Functions

- [ds_list ds_list_create](#) (const bool free_on_delete, void(*destructor)(void *))
Creates a new list.
- void [ds_list_destroy](#) ([ds_list](#) list)
Destroys a list and frees any associated resources.

- void `ds_list_destructor` (void *list)
A list destructor function.
- `ds_list ds_list_append` (`ds_list` list, void *data)
Appends an element to a list.
- void `ds_list_remove_tail` (`ds_list` list)
Removes the last element of a list.
- void `ds_list_remove_all` (`ds_list` list)
Removes all the elements from a list.
- void * `ds_list_element` (`ds_list` list, const size_t index)
Retrieves the data at a specified index.
- size_t `ds_list_length` (`ds_list` list)
Returns the number of elements in a list.
- bool `ds_list_is_empty` (`ds_list` list)
Checks if a list is empty.
- void `ds_list_seek_start` (`ds_list` list)
Sets the current element to the first element of a list.
- void `ds_list_seek_end` (`ds_list` list)
Sets the current element to the last element of a list.
- void * `ds_list_get_next_data` (`ds_list` list)
Returns the next element of the list.
- void * `ds_list_get_prev_data` (`ds_list` list)
Returns the previous element of the list.

5.33.1 Detailed Description

Implementation of generic doubly-linked list data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.33.2 Function Documentation

5.33.2.1 `ds_list ds_list_append (ds_list list, void * element)`

Appends an element to a list.

Parameters

<i>list</i>	The list to which to append.
<i>element</i>	The element to append.

Returns

The same list, or NULL on failure.

5.33.2.2 `ds_list ds_list_create (const bool free_on_delete, void(*)(void *) destructor)`

Creates a new list.

Parameters

<i>free_on_delete</i>	Set to <code>true</code> if the list elements should be destroyed when removed from the list, and when the list itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the list.
<i>destructor</i>	Pointer to a destructor function to use for destroying the list elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created list, or `NULL` on failure.

5.33.2.3 `void ds_list_destroy (ds_list list)`

Destroys a list and frees any associated resources.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

5.33.2.4 `void ds_list_destructor (void * list)`

A list destructor function.

This function may be passed to `ds_list_create()` when creating a list of lists. It calls `ds_list_destroy()`, but the parameter of `ds_list_destroy()` is not compatible with the function signature expected by `ds_list_create()`, so this function provides an appropriate interface.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

5.33.2.5 `void* ds_list_element (ds_list list, const size_t index)`

Retrieves the data at a specified index.

Parameters

<i>list</i>	The list from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

5.33.2.6 `void* ds_list_get_next_data (ds_list list)`

Returns the next element of the list.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the next element, or `NULL` if the end of the list has been reached.

5.33.2.7 void* ds_list_get_prev_data (ds_list list)

Returns the previous element of the list.

This function returns the data of the "current element", and decrements the current element pointer. Subsequent calls to this function will return successively earlier elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the previous element, or `NULL` if the start of the list has been reached.

5.33.2.8 bool ds_list_is_empty (ds_list list)

Checks if a list is empty.

Parameters

<i>list</i>	The list to check.
-------------	--------------------

Returns

`true` if the list is empty, `false` otherwise.

5.33.2.9 size_t ds_list_length (ds_list list)

Returns the number of elements in a list.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

The number of elements in the list.

5.33.2.10 void ds_list_remove_all (ds_list list)

Removes all the elements from a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

5.33.2.11 void ds_list_remove_tail (ds_list list)

Removes the last element of a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

5.33.2.12 void ds_list_seek_end (ds_list list)

Sets the current element to the last element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

5.33.2.13 void ds_list_seek_start (ds_list list)

Sets the current element to the first element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

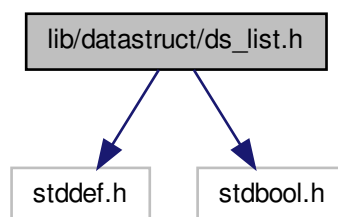
5.34 lib/datastruct/ds_list.h File Reference

Interface to generic doubly-linked list data structure.

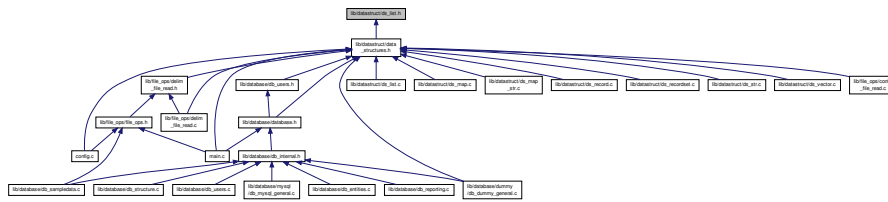
```
#include <stddef.h>
```

```
#include <stdbool.h>
```

Include dependency graph for ds_list.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_list](#) * [ds_list](#)

Functions

- [ds_list ds_list_create](#) (const bool free_on_delete, void(*destructor)(void *))
Creates a new list.
- void [ds_list_destroy](#) ([ds_list](#) list)
Destroys a list and frees any associated resources.
- void [ds_list_destructor](#) (void *list)
A list destructor function.
- [ds_list ds_list_append](#) ([ds_list](#) list, void *element)
Appends an element to a list.
- void [ds_list_remove_tail](#) ([ds_list](#) list)
Removes the last element of a list.
- void [ds_list_remove_all](#) ([ds_list](#) list)
Removes all the elements from a list.
- void * [ds_list_element](#) ([ds_list](#) list, const size_t index)
Retrieves the data at a specified index.
- size_t [ds_list_length](#) ([ds_list](#) list)
Returns the number of elements in a list.
- bool [ds_list_is_empty](#) ([ds_list](#) list)
Checks if a list is empty.
- void [ds_list_seek_start](#) ([ds_list](#) list)
Sets the current element to the first element of a list.
- void [ds_list_seek_end](#) ([ds_list](#) list)
Sets the current element to the last element of a list.
- void * [ds_list_get_next_data](#) ([ds_list](#) list)
Returns the next element of the list.
- void * [ds_list_get_prev_data](#) ([ds_list](#) list)
Returns the previous element of the list.

5.34.1 Detailed Description

Interface to generic doubly-linked list data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.34.2 Typedef Documentation

5.34.2.1 typedef struct ds_list* ds_list

Typedef for opaque list datatype

5.34.3 Function Documentation

5.34.3.1 ds_list ds_list_append (ds_list *list*, void * *element*)

Appends an element to a list.

Parameters

<i>list</i>	The list to which to append.
<i>element</i>	The element to append.

Returns

The same list, or `NULL` on failure.

5.34.3.2 ds_list ds_list_create (const bool *free_on_delete*, void(*) (void *) *destructor*)

Creates a new list.

Parameters

<i>free_on_delete</i>	Set to <code>true</code> if the list elements should be destroyed when removed from the list, and when the list itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the list.
<i>destructor</i>	Pointer to a destructor function to use for destroying the list elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created list, or `NULL` on failure.

5.34.3.3 void ds_list_destroy (ds_list *list*)

Destroys a list and frees any associated resources.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

5.34.3.4 void ds_list_destructor (void * *list*)

A list destructor function.

This function may be passed to `ds_list_create()` when creating a list of lists. It calls `ds_list_destroy()`, but the parameter of `ds_list_destroy()` is not compatible with the function signature expected by `ds_list_create()`, so this function provides an appropriate interface.

Parameters

<i>list</i>	The list to destroy.
-------------	----------------------

5.34.3.5 void* ds_list_element (ds_list *list*, const size_t *index*)

Retrieves the data at a specified index.

Parameters

<i>list</i>	The list from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

5.34.3.6 void* ds_list_get_next_data (ds_list *list*)

Returns the next element of the list.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the next element, or `NULL` if the end of the list has been reached.

5.34.3.7 void* ds_list_get_prev_data (ds_list *list*)

Returns the previous element of the list.

This function returns the data of the "current element", and decrements the current element pointer. Subsequent calls to this function will return successively earlier elements.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

A pointer to the previous element, or `NULL` if the start of the list has been reached.

5.34.3.8 `bool ds_list_is_empty (ds_list list)`

Checks if a list is empty.

Parameters

<i>list</i>	The list to check.
-------------	--------------------

Returns

`true` is the list is empty, `false` otherwise.

5.34.3.9 `size_t ds_list_length (ds_list list)`

Returns the number of elements in a list.

Parameters

<i>list</i>	The list.
-------------	-----------

Returns

The number of elements in the list.

5.34.3.10 `void ds_list_remove_all (ds_list list)`

Removes all the elements from a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

5.34.3.11 `void ds_list_remove_tail (ds_list list)`

Removes the last element of a list.

Parameters

<i>list</i>	The list from which to remove.
-------------	--------------------------------

5.34.3.12 `void ds_list_seek_end (ds_list list)`

Sets the current element to the last element of a list.

Parameters

<i>list</i>	The list.
-------------	-----------

5.34.3.13 `void ds_list_seek_start (ds_list list)`

Sets the current element to the first element of a list.

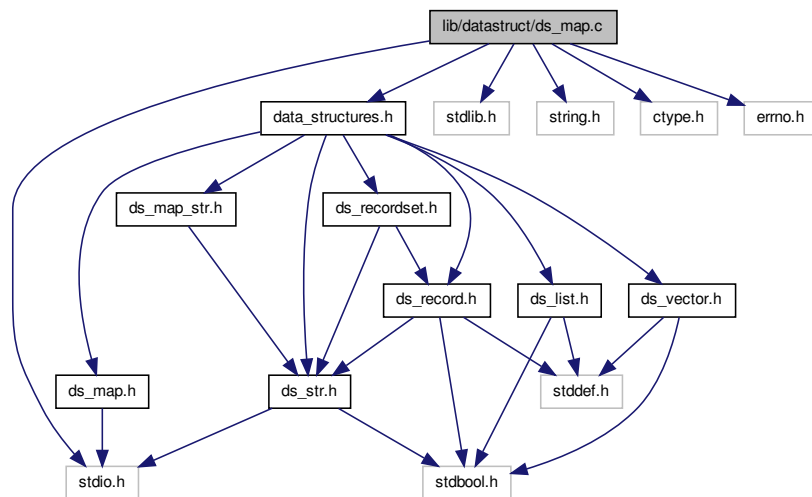
Parameters

<i>list</i>	The list.
-------------	-----------

5.35 lib/datastruct/ds_map.c File Reference

Implementation of string-string hash map data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include "data_structures.h"
Include dependency graph for ds_map.c:
```



Data Structures

- struct [kv_pair_node](#)
- struct [ds_map](#)

Macros

- `#define _POSIX_C_SOURCE 200809L`
Enables POSIX library functions.

Functions

- `ds_map ds_map_init (const size_t hash_size)`
Initializes a hash map.
- `void ds_map_destroy (ds_map map)`
Destroys a hash map.
- `const char * ds_map_get_value (ds_map map, const char *key)`

Retrieves a value associated with a key in the map.

- void `ds_map_insert` (`ds_map` map, const char *key, const char *value)

Inserts a key-value pair into a map.

- void `ds_map_print_all` (`ds_map` map, FILE *outfile)

Prints all the key-value pairs in a map to stdout.

5.35.1 Detailed Description

Implementation of string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.35.2 Function Documentation

5.35.2.1 void `ds_map_destroy` (`ds_map` map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

5.35.2.2 const char* `ds_map_get_value` (`ds_map` map, const char * key)

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

5.35.2.3 `ds_map` `ds_map_init` (const size_t hash_size)

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

5.35.2.4 `void ds_map_insert (ds_map map, const char * key, const char * value)`

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

5.35.2.5 `void ds_map_print_all (ds_map map, FILE * outfile)`

Prints all the key-value pairs in a map to stdout.

Parameters

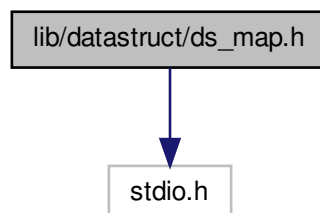
<i>map</i>	A reference to the map.
<i>outfile</i>	A FILE pointer to which to print the output.

5.36 lib/datastruct/ds_map.h File Reference

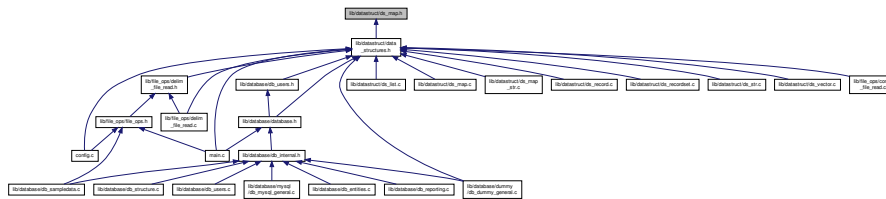
Interface to string-string hash map data structure.

```
#include <stdio.h>
```

Include dependency graph for ds_map.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_map](#) * [ds_map](#)

Functions

- [ds_map ds_map_init](#) (const size_t hash_size)
Initializes a hash map.
- void [ds_map_destroy](#) ([ds_map](#) map)
Destroys a hash map.
- const char * [ds_map_get_value](#) ([ds_map](#) map, const char *key)
Retrieves a value associated with a key in the map.
- void [ds_map_insert](#) ([ds_map](#) map, const char *key, const char *value)
Inserts a key-value pair into a map.
- void [ds_map_print_all](#) ([ds_map](#) map, FILE *outfile)
Prints all the key-value pairs in a map to stdout.

5.36.1 Detailed Description

Interface to string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.36.2 Typedef Documentation

5.36.2.1 typedef struct [ds_map](#)* [ds_map](#)

Opaque data type for hash map

5.36.3 Function Documentation

5.36.3.1 void [ds_map_destroy](#) ([ds_map](#) map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

5.36.3.2 `const char* ds_map_get_value (ds_map map, const char * key)`

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

5.36.3.3 `ds_map ds_map_init (const size_t hash_size)`

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

5.36.3.4 `void ds_map_insert (ds_map map, const char * key, const char * value)`

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

5.36.3.5 `void ds_map_print_all (ds_map map, FILE * outfile)`

Prints all the key-value pairs in a map to stdout.

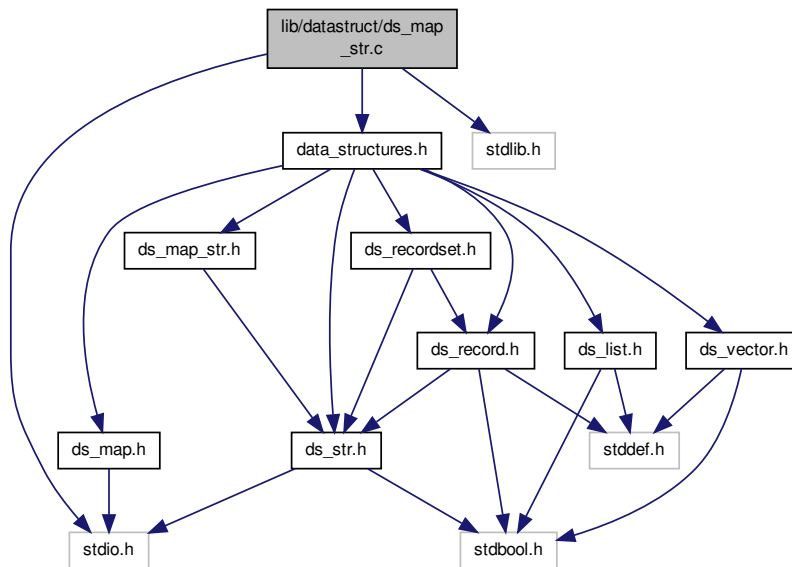
Parameters

<i>map</i>	A reference to the map.
<i>outfile</i>	A FILE pointer to which to print the output.

5.37 lib/datastruct/ds_map_str.c File Reference

Implementation of string-string hash map data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include "data_structures.h"
Include dependency graph for ds_map_str.c:
```



Data Structures

- struct [kv_pair_node](#)
- struct [ds_map_str](#)

Functions

- [ds_map_str ds_map_str_init](#) (const [size_t](#) hash_size)
Initializes a hash map.
- void [ds_map_str_destroy](#) ([ds_map_str](#) map)
Destroys a hash map.
- [ds_str ds_map_str_get_value](#) ([ds_map_str](#) map, [ds_str](#) key)
Retrieves a value associated with a key in the map.
- void [ds_map_str_insert](#) ([ds_map_str](#) map, [ds_str](#) key, [ds_str](#) value)
Inserts a key-value pair into a map.

5.37.1 Detailed Description

Implementation of string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.37.2 Function Documentation

5.37.2.1 void ds_map_str_destroy (ds_map_str map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

5.37.2.2 ds_str ds_map_str_get_value (ds_map_str map, ds_str key)

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

5.37.2.3 ds_map_str ds_map_str_init (const size_t hash_size)

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

5.37.2.4 void ds_map_str_insert (ds_map_str map, ds_str key, ds_str value)

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

Parameters

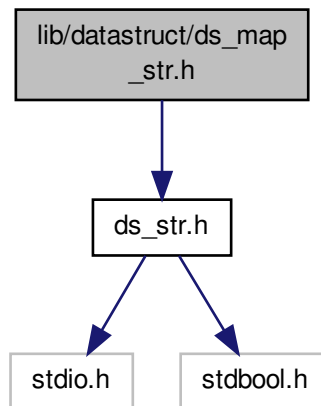
<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

5.38 lib/datastruct/ds_map_str.h File Reference

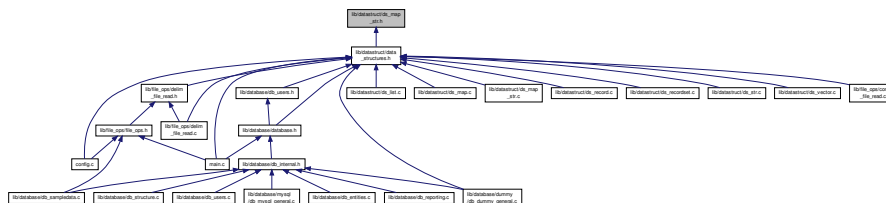
Interface to string-string hash map data structure.

```
#include "ds_str.h"
```

Include dependency graph for ds_map_str.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct `ds_map_str` * `ds_map_str`

Functions

- `ds_map_str ds_map_str_init` (const size_t hash_size)
Initializes a hash map.
- void `ds_map_str_destroy` (`ds_map_str` map)
Destroys a hash map.
- `ds_str ds_map_str_get_value` (`ds_map_str` map, `ds_str` key)
Retrieves a value associated with a key in the map.
- void `ds_map_str_insert` (`ds_map_str` map, `ds_str` key, `ds_str` value)
Inserts a key-value pair into a map.

5.38.1 Detailed Description

Interface to string-string hash map data structure.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.38.2 Typedef Documentation

5.38.2.1 typedef struct ds_map_str* ds_map_str

Opaque data type for hash map

5.38.3 Function Documentation

5.38.3.1 void ds_map_str_destroy (ds_map_str map)

Destroys a hash map.

Parameters

<i>map</i>	A reference to the map to destroy.
------------	------------------------------------

5.38.3.2 ds_str ds_map_str_get_value (ds_map_str map, ds_str key)

Retrieves a value associated with a key in the map.

Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.

Returns

A pointer to the value associated with the key, or `NULL` if the key is not in the map. The caller should not modify the string to which this pointer points.

5.38.3.3 ds_map_str ds_map_str_init (const size_t hash_size)

Initializes a hash map.

Parameters

<i>hash_size</i>	The number of possible hash values.
------------------	-------------------------------------

Returns

A reference to the newly-created hash map.

5.38.3.4 void ds_map_str_insert (ds_map_str map, ds_str key, ds_str value)

Inserts a key-value pair into a map.

The key and value are copied, so the caller may modify or `free()` them after calling this function.

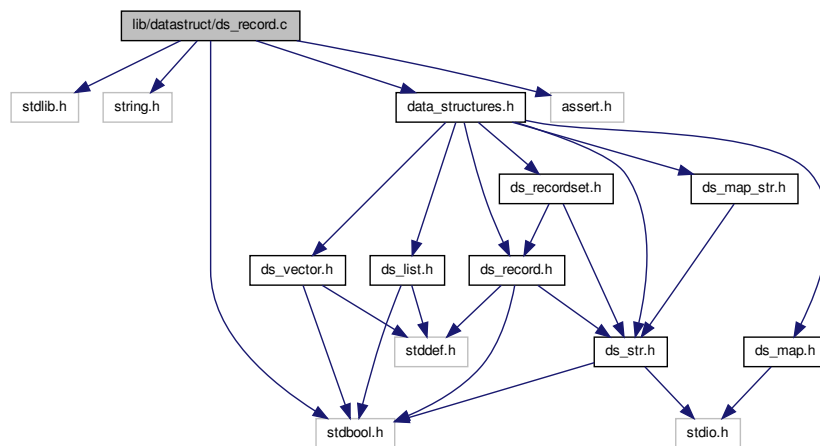
Parameters

<i>map</i>	A reference to the hash map.
<i>key</i>	The key.
<i>value</i>	The value.

5.39 lib/datastruct/ds_record.c File Reference

Implementation of record database structure.

```
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_record.c:
```



Data Structures

- struct [ds_record](#)

Functions

- [ds_record ds_record_create](#) (const size_t size)
Creates a new record.
- void [ds_record_destroy](#) (ds_record record)

- Destroys a record and frees any associated resources.*
- void `ds_record_destructor` (void *record)
A record destructor function.
- void `ds_record_clear` (ds_record record)
Clears and `free()`s all the elements in a record.
- void `ds_record_set_field` (ds_record record, const size_t index, ds_str field)
Sets a field of a record.
- ds_str `ds_record_get_field` (ds_record record, const size_t index)
Retrieves the field at a specified index.
- size_t `ds_record_size` (ds_record record)
Returns the size of a record.
- void `ds_record_seek_start` (ds_record record)
Sets the current field to the first field of a record.
- ds_str `ds_record_get_next_data` (ds_record record)
Returns the next field of the record.
- ds_record `ds_record_tokenize` (ds_str str, const char delim)
Tokenizes a string into a record.
- ds_str `ds_record_make_delim_string` (ds_record record, const char delim)
Makes a delimited string from a record.
- ds_str `ds_record_make_values_string` (ds_record record)
Makes a delimited SQL values string from a record.

5.39.1 Detailed Description

Implementation of record database structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.39.2 Function Documentation

5.39.2.1 void ds_record_clear (ds_record record)

Clears and `free()`s all the elements in a record.

Parameters

<i>record</i>	The record.
---------------	-------------

5.39.2.2 ds_record ds_record_create (const size_t size)

Creates a new record.

Parameters

<i>size</i>	The size of the record.
-------------	-------------------------

Returns

A newly created record, or `NULL` on failure.

5.39.2.3 void ds_record_destroy (ds_record record)

Destroys a record and frees any associated resources.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

5.39.2.4 void ds_record_destructor (void * record)

A record destructor function.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

5.39.2.5 ds_str ds_record_get_field (ds_record record, const size_t index)

Retrieves the field at a specified index.

Parameters

<i>record</i>	The record from which to retrieve.
<i>index</i>	The index of the desired field.

Returns

A pointer to the field, or `NULL` if the index is out of range.

5.39.2.6 ds_str ds_record_get_next_data (ds_record record)

Returns the next field of the record.

This function returns the data of the "current field", and advances the current field pointer. Subsequent calls to this function will return successive fields.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

A pointer to the next field, or `NULL` if the end of the record has been reached.

5.39.2.7 ds_str ds_record_make_delim_string (ds_record record, const char delim)

Makes a delimited string from a record.

Parameters

<i>record</i>	The record.
<i>delim</i>	The delimiting character.

Returns

The delimited string, or `NULL` on failure.

5.39.2.8 ds_str ds_record_make_values_string (ds_record record)

Makes a delimited SQL values string from a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The delimited values string, or `NULL` on failure.

5.39.2.9 void ds_record_seek_start (ds_record record)

Sets the current field to the first field of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

5.39.2.10 void ds_record_set_field (ds_record record, const size_t index, ds_str field)

Sets a field of a record.

If the field is currently occupied, the existing field is `free()`d.

Parameters

<i>record</i>	The record to set.
<i>index</i>	The index of the field to set.
<i>field</i>	The value to which to set the field.

5.39.2.11 size_t ds_record_size (ds_record record)

Returns the size of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The size of the record.

5.39.2.12 `ds_record ds_record_tokenize (ds_str str, const char delim)`

Tokenizes a string into a record.

Parameters

<i>str</i>	The string to tokenize.
<i>delim</i>	The delimiting character.

Returns

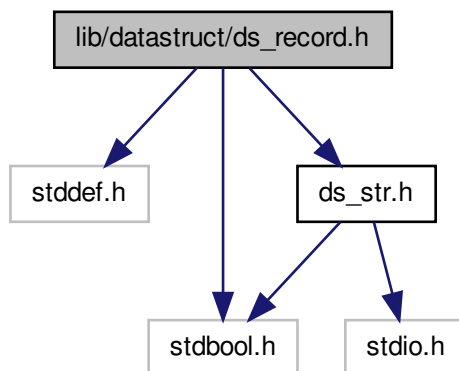
A new record containing the tokens.

5.40 `lib/datastruct/ds_record.h` File Reference

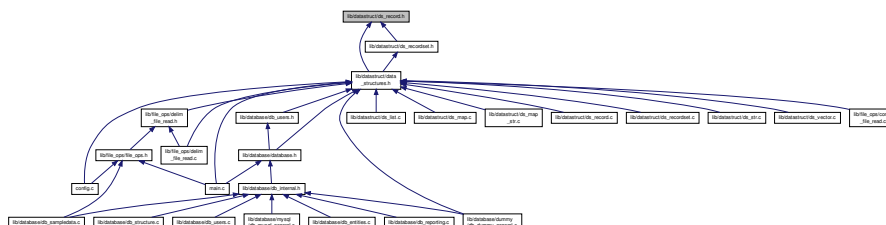
Interface to record data structure.

```
#include <stddef.h>
#include <stdbool.h>
#include "ds_str.h"
```

Include dependency graph for `ds_record.h`:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct `ds_record` * `ds_record`

Functions

- [ds_record ds_record_create](#) (const size_t size)
Creates a new record.
- void [ds_record_destroy](#) ([ds_record](#) record)
Destroys a record and frees any associated resources.
- void [ds_record_destructor](#) (void *record)
A record destructor function.
- void [ds_record_clear](#) ([ds_record](#) record)
Clears and `free()`s all the elements in a record.
- void [ds_record_set_field](#) ([ds_record](#) record, const size_t index, [ds_str](#) field)
Sets a field of a record.
- [ds_str ds_record_get_field](#) ([ds_record](#) record, const size_t index)
Retrieves the field at a specified index.
- size_t [ds_record_size](#) ([ds_record](#) record)
Returns the size of a record.
- void [ds_record_seek_start](#) ([ds_record](#) record)
Sets the current field to the first field of a record.
- [ds_str ds_record_get_next_data](#) ([ds_record](#) record)
Returns the next field of the record.
- [ds_record ds_record_tokenize](#) ([ds_str](#) str, const char delim)
Tokenizes a string into a record.
- [ds_str ds_record_make_delim_string](#) ([ds_record](#) record, const char delim)
Makes a delimited string from a record.
- [ds_str ds_record_make_values_string](#) ([ds_record](#) record)
Makes a delimited SQL values string from a record.

5.40.1 Detailed Description

Interface to record data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.40.2 Typedef Documentation

5.40.2.1 typedef struct ds_record* ds_record

Typedef for opaque record datatype

5.40.3 Function Documentation

5.40.3.1 void ds_record_clear (ds_record record)

Clears and `free()`s all the elements in a record.

Parameters

<i>record</i>	The record.
---------------	-------------

5.40.3.2 `ds_record ds_record_create (const size_t size)`

Creates a new record.

Parameters

<i>size</i>	The size of the record.
-------------	-------------------------

Returns

A newly created record, or `NULL` on failure.

5.40.3.3 `void ds_record_destroy (ds_record record)`

Destroys a record and frees any associated resources.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

5.40.3.4 `void ds_record_destructor (void * record)`

A record destructor function.

Parameters

<i>record</i>	The record to destroy.
---------------	------------------------

5.40.3.5 `ds_str ds_record_get_field (ds_record record, const size_t index)`

Retrieves the field at a specified index.

Parameters

<i>record</i>	The record from which to retrieve.
<i>index</i>	The index of the desired field.

Returns

A pointer to the field, or `NULL` if the index is out of range.

5.40.3.6 `ds_str ds_record_get_next_data (ds_record record)`

Returns the next field of the record.

This function returns the data of the "current field", and advances the current field pointer. Subsequent calls to this function will return successive fields.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

A pointer to the next field, or `NULL` if the end of the record has been reached.

5.40.3.7 ds_str ds_record_make_delim_string (ds_record record, const char delim)

Makes a delimited string from a record.

Parameters

<i>record</i>	The record.
<i>delim</i>	The delimiting character.

Returns

The delimited string, or `NULL` on failure.

5.40.3.8 ds_str ds_record_make_values_string (ds_record record)

Makes a delimited SQL values string from a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The delimited values string, or `NULL` on failure.

5.40.3.9 void ds_record_seek_start (ds_record record)

Sets the current field to the first field of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

5.40.3.10 void ds_record_set_field (ds_record record, const size_t index, ds_str field)

Sets a field of a record.

If the field is currently occupied, the existing field is `free()`d.

Parameters

<i>record</i>	The record to set.
<i>index</i>	The index of the field to set.
<i>field</i>	The value to which to set the field.

5.40.3.11 `size_t ds_record_size (ds_record record)`

Returns the size of a record.

Parameters

<i>record</i>	The record.
---------------	-------------

Returns

The size of the record.

5.40.3.12 `ds_record ds_record_tokenize (ds_str str, const char delim)`

Tokenizes a string into a record.

Parameters

<i>str</i>	The string to tokenize.
<i>delim</i>	The delimiting character.

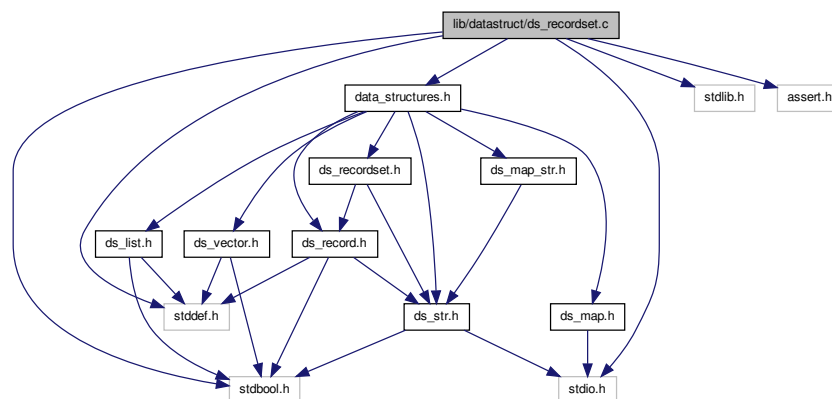
Returns

A new record containing the tokens.

5.41 `lib/datastruct/ds_recordset.c` File Reference

Implementation of query result set structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_recordset.c:
```



Data Structures

- struct [ds_recordset](#)

Functions

- [ds_recordset ds_recordset_create](#) (const size_t num_fields)
Creates a new record set.
- void [ds_recordset_destroy](#) (ds_recordset set)
Destroys a record set and frees associated resources.
- [ds_record ds_recordset_add_record](#) (ds_recordset set, ds_record record)
Adds a record to a record set.
- size_t [ds_recordset_num_fields](#) (ds_recordset set)
Returns the number of fields in a record set.
- size_t [ds_recordset_num_records](#) (ds_recordset set)
Returns the number of records in a record set.
- void [ds_recordset_set_headers](#) (ds_recordset set, ds_record headers)
Sets the record headers in a record set.
- [ds_str ds_recordset_get_text_report](#) (ds_recordset set)
Returns a formatted text report for the record set.
- void [ds_recordset_seek_start](#) (ds_recordset set)
Sets the current record to the first record.
- [ds_record ds_recordset_next_record](#) (ds_recordset set)
Returns the next record in the record set.
- [ds_str ds_recordset_get_next_insert_query](#) (ds_recordset set, const char *table_name)
Gets the next SQL INSERT query.

5.41.1 Detailed Description

Implementation of query result set structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.41.2 Function Documentation

5.41.2.1 ds_record ds_recordset_add_record (ds_recordset set, ds_record record)

Adds a record to a record set.

The record *must* have the same number of fields as the number of fields provided to [ds_recordset_create\(\)](#).

Parameters

<i>set</i>	The record set to which to add.
<i>record</i>	The record to add.

Returns

A pointer to the new record (i.e. it returns the second parameter) or `NULL` on failure.

5.41.2.2 `ds_recordset ds_recordset_create (const size_t num_fields)`

Creates a new record set.

Parameters

<i>num_fields</i>	The non-zero number of fields in the record set.
-------------------	--

Returns

A pointer to the new record set.

5.41.2.3 `void ds_recordset_destroy (ds_recordset set)`

Destroys a record set and frees associated resources.

Parameters

<i>set</i>	The record set to destroy.
------------	----------------------------

5.41.2.4 `ds_str ds_recordset_get_next_insert_query (ds_recordset set, const char * table_name)`

Gets the next SQL INSERT query.

Parameters

<i>set</i>	The set.
<i>table_name</i>	The table name into which to insert.

Returns

The query. Caller is responsible for `free()` ing.

5.41.2.5 `ds_str ds_recordset_get_text_report (ds_recordset set)`

Returns a formatted text report for the record set.

The report is returned as a single multi-line string.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the report. The caller is responsible for `free()` ing this pointer.

5.41.2.6 `ds_record ds_recordset_next_record (ds_recordset set)`

Returns the next record in the record set.

This function returns the "current record", and advances the current record pointer. Subsequent calls to this function will return successive records.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the next record, or `NULL` if the end of the record set has been reached.

5.41.2.7 `size_t ds_recordset_num_fields (ds_recordset set)`

Returns the number of fields in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of fields in the record set.

5.41.2.8 `size_t ds_recordset_num_records (ds_recordset set)`

Returns the number of records in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of records in the record set.

5.41.2.9 `void ds_recordset_seek_start (ds_recordset set)`

Sets the current record to the first record.

Parameters

<i>set</i>	The record set.
------------	-----------------

5.41.2.10 `void ds_recordset_set_headers (ds_recordset set, ds_record headers)`

Sets the record headers in a record set.

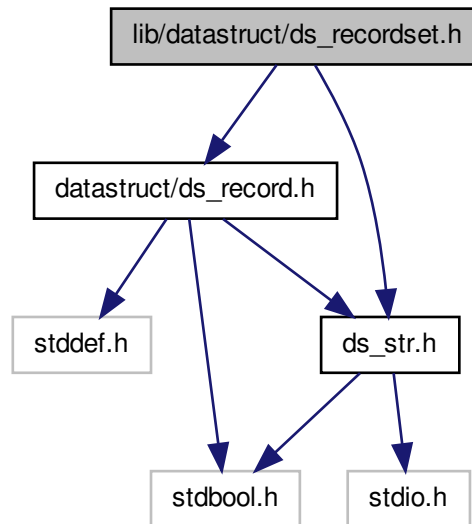
Parameters

<i>set</i>	The record set.
<i>headers</i>	The headers, in the form of a ds_record of strings. The list <i>must</i> have the same number of elements as the number of fields provided to ds_recordset_create() .

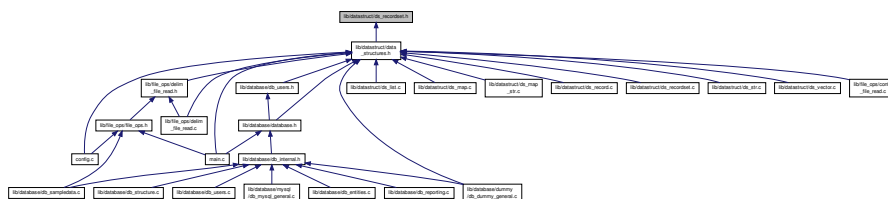
5.42 lib/datastruct/ds_recordset.h File Reference

Interface to record set structure.

```
#include "datastruct/ds_record.h"
#include "datastruct/ds_str.h"
Include dependency graph for ds_recordset.h:
```



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_recordset](#) * [ds_recordset](#)

Functions

- [ds_recordset ds_recordset_create](#) (const size_t num_fields)
Creates a new record set.
- void [ds_recordset_destroy](#) ([ds_recordset](#) set)
Destroys a record set and frees associated resources.
- [ds_record ds_recordset_add_record](#) ([ds_recordset](#) set, [ds_record](#) record)

- Adds a record to a record set.*
- size_t `ds_recordset_num_fields` (`ds_recordset` set)
Returns the number of fields in a record set.
- size_t `ds_recordset_num_records` (`ds_recordset` set)
Returns the number of records in a record set.
- void `ds_recordset_set_headers` (`ds_recordset` set, `ds_record` headers)
Sets the record headers in a record set.
- `ds_str` `ds_recordset_get_text_report` (`ds_recordset` set)
Returns a formatted text report for the record set.
- `ds_str` `ds_recordset_get_next_insert_query` (`ds_recordset` set, const char *table_name)
Gets the next SQL INSERT query.
- void `ds_recordset_seek_start` (`ds_recordset` set)
Sets the current record to the first record.
- `ds_record` `ds_recordset_next_record` (`ds_recordset` set)
Returns the next record in the record set.

5.42.1 Detailed Description

Interface to record set structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.42.2 Typedef Documentation

5.42.2.1 typedef struct `ds_recordset*` `ds_recordset`

Typedef for opaque record set data type

5.42.3 Function Documentation

5.42.3.1 `ds_record` `ds_recordset_add_record` (`ds_recordset` set, `ds_record` record)

Adds a record to a record set.

The record *must* have the same number of fields as the number of fields provided to `ds_recordset_create()`.

Parameters

<i>set</i>	The record set to which to add.
<i>record</i>	The record to add.

Returns

A pointer to the new record (i.e. it returns the second parameter) or `NULL` on failure.

5.42.3.2 `ds_recordset ds_recordset_create (const size_t num_fields)`

Creates a new record set.

Parameters

<i>num_fields</i>	The non-zero number of fields in the record set.
-------------------	--

Returns

A pointer to the new record set.

5.42.3.3 `void ds_recordset_destroy (ds_recordset set)`

Destroys a record set and frees associated resources.

Parameters

<i>set</i>	The record set to destroy.
------------	----------------------------

5.42.3.4 `ds_str ds_recordset_get_next_insert_query (ds_recordset set, const char * table_name)`

Gets the next SQL INSERT query.

Parameters

<i>set</i>	The set.
<i>table_name</i>	The table name into which to insert.

Returns

The query. Caller is responsible for `free()` ing.

5.42.3.5 `ds_str ds_recordset_get_text_report (ds_recordset set)`

Returns a formatted text report for the record set.

The report is returned as a single multi-line string.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the report. The caller is responsible for `free()` ing this pointer.

5.42.3.6 `ds_record ds_recordset_next_record (ds_recordset set)`

Returns the next record in the record set.

This function returns the "current record", and advances the current record pointer. Subsequent calls to this function will return successive records.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

A pointer to the next record, or `NULL` if the end of the record set has been reached.

5.42.3.7 `size_t ds_recordset_num_fields (ds_recordset set)`

Returns the number of fields in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of fields in the record set.

5.42.3.8 `size_t ds_recordset_num_records (ds_recordset set)`

Returns the number of records in a record set.

Parameters

<i>set</i>	The record set.
------------	-----------------

Returns

The number of records in the record set.

5.42.3.9 `void ds_recordset_seek_start (ds_recordset set)`

Sets the current record to the first record.

Parameters

<i>set</i>	The record set.
------------	-----------------

5.42.3.10 `void ds_recordset_set_headers (ds_recordset set, ds_record headers)`

Sets the record headers in a record set.

Parameters

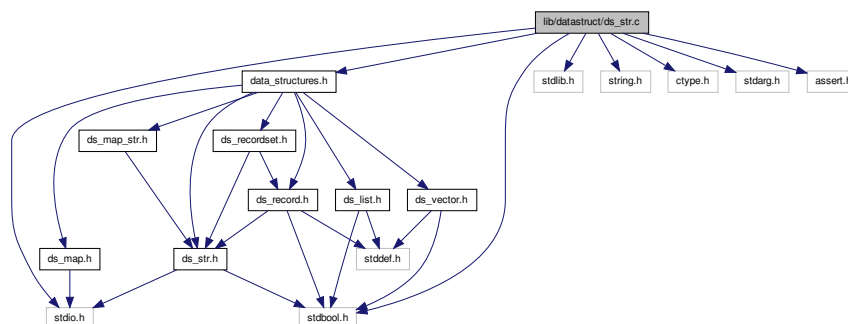
<i>set</i>	The record set.
<i>headers</i>	The headers, in the form of a ds_record of strings. The list <i>must</i> have the same number of elements as the number of fields provided to ds_recordset_create() .

5.43 lib/datastruct/ds_str.c File Reference

Implementation of string data structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>
#include <assert.h>
#include "data_structures.h"
```

Include dependency graph for ds_str.c:



Data Structures

- struct [ds_str](#)

Functions

- [ds_str ds_str_create_direct](#) (char *init_str, const size_t init_str_size)
Creates a string using allocated memory.
- [ds_str ds_str_create](#) (const char *init_str)
Creates a new string from a C-style string.
- [ds_str ds_str_dup](#) (ds_str src)
Creates a new string from another string.
- [ds_str ds_str_create_sprintf](#) (const char *format,...)
Creates a string with `sprintf()`-type format.
- void [ds_str_destroy](#) (ds_str str)
Destroys a string and releases allocated resources.
- void [ds_str_destructor](#) (void *str)
Destroys a string and releases allocated resources.
- [ds_str ds_str_assign](#) (ds_str dst, ds_str src)
Assigns a string to another.
- [ds_str ds_str_assign_cstr](#) (ds_str dst, const char *src)
Assigns a C-style string to a string.
- const char * [ds_str_cstr](#) (ds_str str)
Returns a C-style string containing the string's contents.
- size_t [ds_str_length](#) (ds_str str)

- Returns the length of a string.*

 - `ds_str ds_str_size_to_fit (ds_str str)`

Reduces a string's capacity to fit its length.
- `ds_str ds_str_concat (ds_str dst, ds_str src)`

Concatenates two strings.
- `ds_str ds_str_concat_cstr (ds_str dst, const char *src)`

Concatenates a C-style string to a string.
- `ds_str ds_str_trunc (ds_str str, const size_t length)`

Truncates a string.
- unsigned long `ds_str_hash (ds_str str)`

Calculates a hash of a string.
- int `ds_str_compare (ds_str s1, ds_str s2)`

Compares two strings.
- int `ds_str_compare_cstr (ds_str s1, const char *s2)`

Compares a string with a C-style string.
- int `ds_str_strchr (ds_str str, const char ch, const int start)`

Returns index of first occurrence of a character.
- `ds_str ds_str_substr_left (ds_str str, const size_t numchars)`

Returns a left substring.
- `ds_str ds_str_substr_right (ds_str str, const size_t numchars)`

Returns a right substring.
- void `ds_str_split (ds_str src, ds_str *left, ds_str *right, const char sc)`

Splits a string.
- void `ds_str_trim_leading (ds_str str)`

Trims leading whitespace in-place.
- void `ds_str_trim_trailing (ds_str str)`

Trims trailing whitespace in-place.
- void `ds_str_trim (ds_str str)`

Trims leading and trailing whitespace in-place.
- char `ds_str_char_at_index (ds_str str, const size_t index)`

Returns the character at a specified index.
- bool `ds_str_is_empty (ds_str str)`

Checks if a string is empty.
- void `ds_str_clear (ds_str str)`

Clears (empties) a string.
- bool `ds_str_intval (ds_str str, const int base, int *value)`

Gets the integer value of a string.
- bool `ds_str_doubleval (ds_str str, double *value)`

Gets the double value of a string.
- `ds_str ds_str_getline (ds_str str, const size_t size, FILE *fp)`

Gets a line from a file and assigns it to a string.
- `ds_str ds_str_decorate (ds_str str, ds_str left_dec, ds_str right_dec)`

Brackets a string with decoration strings.

5.43.1 Detailed Description

Implementation of string data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.43.2 Function Documentation

5.43.2.1 `ds_str ds_str_assign (ds_str dst, ds_str src)`

Assigns a string to another.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source string.

Returns

dst on success, `NULL` on failure.

5.43.2.2 `ds_str ds_str_assign_cstr (ds_str dst, const char * src)`

Assigns a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source C-style string.

Returns

dst on success, `NULL` on failure.

5.43.2.3 `char ds_str_char_at_index (ds_str str, const size_t index)`

Returns the character at a specified index.

Parameters

<i>str</i>	The string.
<i>index</i>	The specified index.

Returns

The character at the specified index.

5.43.2.4 void ds_str_clear (ds_str str)

Clears (empties) a string.

Parameters

<i>str</i>	The string.
------------	-------------

5.43.2.5 int ds_str_compare (ds_str s1, ds_str s2)

Compares two strings.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second string.

Returns

Less than, equal to, or greater than zero if s1 is found, respectively, to be less than, equal to, or greater than s2.

5.43.2.6 int ds_str_compare_cstr (ds_str s1, const char * s2)

Compares a string with a C-style string.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second, C-Style string.

Returns

Less than, equal to, or greater than zero if s1 is found, respectively, to be less than, equal to, or greater than s2.

5.43.2.7 ds_str ds_str_concat (ds_str dst, ds_str src)

Concatenates two strings.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or NULL on failure.

5.43.2.8 ds_str ds_str_concat_cstr (ds_str dst, const char * src)

Concatenates a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or `NULL` on failure.

5.43.2.9 `ds_str ds_str_create (const char * init_str)`

Creates a new string from a C-style string.

Parameters

<i>init_str</i>	The C-style string.
-----------------	---------------------

Returns

The new string, or `NULL` on failure.

5.43.2.10 `ds_str ds_str_create_direct (char * init_str, const size_t init_str_size)`

Creates a string using allocated memory.

The normal construction functions duplicate the string used to create it. In cases where allocated memory is already available (e.g. in `ds_str_create_sprintf()`) this function allows that memory to be directly assigned to the string, avoiding an unnecessary duplication.

Parameters

<i>init_str</i>	The allocated memory. IMPORTANT: If the construction of the string fails, this memory will be <code>free()</code> d.
<i>init_str_size</i>	The size of the allocated memory. IMPORTANT: The string's length is assumed to be one less than this quantity, and a call to <code>strlen()</code> is NOT performed.

Returns

The new string, or `NULL` on failure.

5.43.2.11 `ds_str ds_str_create_sprintf (const char * format, ...)`

Creates a string with `sprintf()`-type format.

Parameters

<i>format</i>	The format string.
<i>...</i>	The subsequent arguments as specified by the format string.

Returns

The new string, or `NULL` on failure.

5.43.2.12 `const char* ds_str_cstr (ds_str str)`

Returns a C-style string containing the string's contents.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The C-style string containing the string's contents. The caller should not directly modify this string.

5.43.2.13 `ds_str ds_str_decorate (ds_str str, ds_str left_dec, ds_str right_dec)`

Brackets a string with decoration strings.

Parameters

<i>str</i>	The string to decorate.
<i>left_dec</i>	The string to add to the left of <i>str</i> .
<i>right_dec</i>	The string to add to the right of <i>str</i> , or NULL to add <i>left_dec</i> to both sides.

Returns

The decorated string.

5.43.2.14 `void ds_str_destroy (ds_str str)`

Destroys a string and releases allocated resources.

Parameters

<i>str</i>	The string to destroy..
------------	-------------------------

5.43.2.15 `void ds_str_destructor (void * str)`

Destroys a string and releases allocated resources.

This function calls `ds_str_destroy()`, and can be passed to a data structure expecting a destructor function with the signature `void (*)(void *)`.

Parameters

<i>str</i>	The string to destroy.
------------	------------------------

5.43.2.16 `bool ds_str_doubleval (ds_str str, double * value)`

Gets the double value of a string.

Parameters

<i>str</i>	The string.
<i>value</i>	A pointer to the double in which to store the value. Zero is stored if the string does not contain a valid double value.

Returns

`true` on successful conversion, `false` if the string does not contain a valid double value.

5.43.2.17 `ds_str ds_str_dup (ds_str src)`

Creates a new string from another string.

Parameters

<i>src</i>	The other string.
------------	-------------------

Returns

The new string, or `NULL` on failure.

5.43.2.18 `ds_str ds_str_getline (ds_str str, const size_t size, FILE * fp)`

Gets a line from a file and assigns it to a string.

Any trailing newline character is stripped.

Parameters

<i>str</i>	The string.
<i>size</i>	The maximum number of bytes to read, including the null.
<i>fp</i>	The file pointer from which to read.

Returns

`dst`

5.43.2.19 `unsigned long ds_str_hash (ds_str str)`

Calculates a hash of a string.

Uses Dan Bernstein's djb2 algorithm.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The hash value

5.43.2.20 `bool ds_str_intval (ds_str str, const int base, int * value)`

Gets the integer value of a string.

Parameters

<i>str</i>	The string.
<i>base</i>	The base of the integer. This has the same meaning as the third argument to standard C <code>strtol()</code> .

<i>value</i>	A pointer to the integer in which to store the value. Zero is stored if the string does not contain a valid integer value.
--------------	--

Returns

`true` on successful conversion, `false` if the string does not contain a valid integer value.

5.43.2.21 `bool ds_str_is_empty (ds_str str)`

Checks if a string is empty.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

`true` if the string is empty, `false` otherwise.

5.43.2.22 `size_t ds_str_length (ds_str str)`

Returns the length of a string.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The length of the string.

5.43.2.23 `ds_str ds_str_size_to_fit (ds_str str)`

Reduces a string's capacity to fit its length.

Parameters

<i>str</i>	The string to size.
------------	---------------------

Returns

`str`, or `NULL` on failure.

5.43.2.24 `void ds_str_split (ds_str src, ds_str * left, ds_str * right, const char sc)`

Splits a string.

Parameters

<i>src</i>	The string to split.
<i>left</i>	Pointer to left substring (modified)
<i>right</i>	Pointer to right substring (modified)
<i>sc</i>	Split character.

5.43.2.25 `int ds_str_strchr (ds_str str, const char ch, const int start)`

Returns index of first occurrence of a character.

Parameters

<i>str</i>	The string.
<i>ch</i>	The character for which to search.
<i>start</i>	The index of the string at which to start looking. Set this to non-zero to begin searching from a point other than the first character of the string.

Returns

The index of the first occurrence, or -1 if the character was not found.

5.43.2.26 `ds_str ds_str_substr_left (ds_str str, const size_t numchars)`

Returns a left substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of left characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

5.43.2.27 `ds_str ds_str_substr_right (ds_str str, const size_t numchars)`

Returns a right substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of right characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

5.43.2.28 `void ds_str_trim (ds_str str)`

Trims leading and trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

5.43.2.29 void ds_str_trim_leading (ds_str str)

Trims leading whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

5.43.2.30 void ds_str_trim_trailing (ds_str str)

Trims trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

5.43.2.31 ds_str ds_str_trunc (ds_str str, const size_t length)

Truncates a string.

Parameters

<i>str</i>	The string.
<i>length</i>	The new length to which to truncate.

Returns

The original string, or `NULL` on failure.

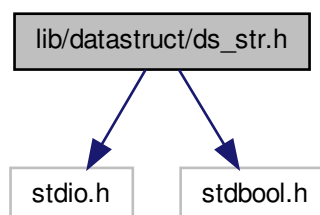
5.44 lib/datastruct/ds_str.h File Reference

Interface to string data structure.

```
#include <stdio.h>
```

```
#include <stdbool.h>
```

Include dependency graph for ds_str.h:



- int `ds_str_strchr` (`ds_str` str, const char ch, const int start)
Returns index of first occurrence of a character.
- `ds_str ds_str_substr_left` (`ds_str` str, const size_t numchars)
Returns a left substring.
- `ds_str ds_str_substr_right` (`ds_str` str, const size_t numchars)
Returns a right substring.
- void `ds_str_split` (`ds_str` src, `ds_str` *left, `ds_str` *right, const char sc)
Splits a string.
- void `ds_str_trim_leading` (`ds_str` str)
Trims leading whitespace in-place.
- void `ds_str_trim_trailing` (`ds_str` str)
Trims trailing whitespace in-place.
- void `ds_str_trim` (`ds_str` str)
Trims leading and trailing whitespace in-place.
- char `ds_str_char_at_index` (`ds_str` str, const size_t index)
Returns the character at a specified index.
- bool `ds_str_is_empty` (`ds_str` str)
Checks if a string is empty.
- void `ds_str_clear` (`ds_str` str)
Clears (empties) a string.
- bool `ds_str_intval` (`ds_str` str, const int base, int *value)
Gets the integer value of a string.
- bool `ds_str_doubleval` (`ds_str` str, double *value)
Gets the double value of a string.
- `ds_str ds_str_getline` (`ds_str` str, const size_t size, FILE *fp)
Gets a line from a file and assigns it to a string.
- `ds_str ds_str_decorate` (`ds_str` str, `ds_str` left_dec, `ds_str` right_dec)
Brackets a string with decoration strings.

5.44.1 Detailed Description

Interface to string data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.44.2 Typedef Documentation

5.44.2.1 typedef struct ds_str* ds_str

Opaque data type for string

5.44.3 Function Documentation

5.44.3.1 `ds_str ds_str_assign (ds_str dst, ds_str src)`

Assigns a string to another.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source string.

Returns

dst on success, `NULL` on failure.

5.44.3.2 `ds_str ds_str_assign_cstr (ds_str dst, const char * src)`

Assigns a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source C-style string.

Returns

dst on success, `NULL` on failure.

5.44.3.3 `char ds_str_char_at_index (ds_str str, const size_t index)`

Returns the character at a specified index.

Parameters

<i>str</i>	The string.
<i>index</i>	The specified index.

Returns

The character at the specified index.

5.44.3.4 `void ds_str_clear (ds_str str)`

Clears (empties) a string.

Parameters

<i>str</i>	The string.
------------	-------------

5.44.3.5 `int ds_str_compare (ds_str s1, ds_str s2)`

Compares two strings.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second string.

Returns

Less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, equal to, or greater than *s2*.

5.44.3.6 int ds_str_compare_cstr (ds_str *s1*, const char * *s2*)

Compares a string with a C-style string.

Parameters

<i>s1</i>	The first string.
<i>s2</i>	The second, C-Style string.

Returns

Less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, equal to, or greater than *s2*.

5.44.3.7 ds_str ds_str_concat (ds_str *dst*, ds_str *src*)

Concatenates two strings.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or `NULL` on failure.

5.44.3.8 ds_str ds_str_concat_cstr (ds_str *dst*, const char * *src*)

Concatenates a C-style string to a string.

Parameters

<i>dst</i>	The destination string.
<i>src</i>	The source strings.

Returns

The destination string, or `NULL` on failure.

5.44.3.9 ds_str ds_str_create (const char * *init_str*)

Creates a new string from a C-style string.

Parameters

<i>init_str</i>	The C-style string.
-----------------	---------------------

Returns

The new string, or `NULL` on failure.

5.44.3.10 `ds_str ds_str_create_direct (char * init_str, const size_t init_str_size)`

Creates a string using allocated memory.

The normal construction functions duplicate the string used to create it. In cases where allocated memory is already available (e.g. in `ds_str_create_sprintf()`) this function allows that memory to be directly assigned to the string, avoiding an unnecessary duplication.

Parameters

<i>init_str</i>	The allocated memory. IMPORTANT: If the construction of the string fails, this memory will be <code>free()</code> d.
<i>init_str_size</i>	The size of the allocated memory. IMPORTANT: The string's length is assumed to be one less than this quantity, and a call to <code>strlen()</code> is NOT performed.

Returns

The new string, or `NULL` on failure.

5.44.3.11 `ds_str ds_str_create_sprintf (const char * format, ...)`

Creates a string with `sprintf()`-type format.

Parameters

<i>format</i>	The format string.
<i>...</i>	The subsequent arguments as specified by the format string.

Returns

The new string, or `NULL` on failure.

5.44.3.12 `const char* ds_str_cstr (ds_str str)`

Returns a C-style string containing the string's contents.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The C-style string containing the string's contents. The caller should not directly modify this string.

5.44.3.13 `ds_str ds_str_decorate (ds_str str, ds_str left_dec, ds_str right_dec)`

Brackets a string with decoration strings.

Parameters

<i>str</i>	The string to decorate.
<i>left_dec</i>	The string to add to the left of <i>str</i> .
<i>right_dec</i>	The string to add to the right of <i>str</i> , or <code>NULL</code> to add <i>left_dec</i> to both sides.

Returns

The decorated string.

5.44.3.14 `void ds_str_destroy (ds_str str)`

Destroys a string and releases allocated resources.

Parameters

<i>str</i>	The string to destroy..
------------	-------------------------

5.44.3.15 `void ds_str_destructor (void * str)`

Destroys a string and releases allocated resources.

This function calls `ds_str_destroy()`, and can be passed to a data structure expecting a destructor function with the signature `void (*)(void *)`.

Parameters

<i>str</i>	The string to destroy.
------------	------------------------

5.44.3.16 `bool ds_str_doubleval (ds_str str, double * value)`

Gets the double value of a string.

Parameters

<i>str</i>	The string.
<i>value</i>	A pointer to the double in which to store the value. Zero is stored if the string does not contain a valid double value.

Returns

`true` on successful conversion, `false` if the string does not contain a valid double value.

5.44.3.17 `ds_str ds_str_dup (ds_str src)`

Creates a new string from another string.

Parameters

<i>src</i>	The other string.
------------	-------------------

Returns

The new string, or `NULL` on failure.

5.44.3.18 ds_str ds_str_getline (ds_str str, const size_t size, FILE * fp)

Gets a line from a file and assigns it to a string.

Any trailing newline character is stripped.

Parameters

<i>str</i>	The string.
<i>size</i>	The maximum number of bytes to read, including the null.
<i>fp</i>	The file pointer from which to read.

Returns

`dst`

5.44.3.19 unsigned long ds_str_hash (ds_str str)

Calculates a hash of a string.

Uses Dan Bernstein's djb2 algorithm.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The hash value

5.44.3.20 bool ds_str_intval (ds_str str, const int base, int * value)

Gets the integer value of a string.

Parameters

<i>str</i>	The string.
<i>base</i>	The base of the integer. This has the same meaning as the third argument to standard C <code>strtol()</code> .
<i>value</i>	A pointer to the integer in which to store the value. Zero is stored if the string does not contain a valid integer value.

Returns

`true` on successful conversion, `false` if the string does not contain a valid integer value.

5.44.3.21 `bool ds_str_is_empty (ds_str str)`

Checks if a string is empty.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

`true` is the string is empty, `false` otherwise.

5.44.3.22 `size_t ds_str_length (ds_str str)`

Returns the length of a string.

Parameters

<i>str</i>	The string.
------------	-------------

Returns

The length of the string.

5.44.3.23 `ds_str ds_str_size_to_fit (ds_str str)`

Reduces a string's capacity to fit its length.

Parameters

<i>str</i>	The string to size.
------------	---------------------

Returns

`str`, or `NULL` on failure.

5.44.3.24 `void ds_str_split (ds_str src, ds_str * left, ds_str * right, const char sc)`

Splits a string.

Parameters

<i>src</i>	The string to split.
<i>left</i>	Pointer to left substring (modified)
<i>right</i>	Pointer to right substring (modified)
<i>sc</i>	Split character.

5.44.3.25 `int ds_str_strchr (ds_str str, const char ch, const int start)`

Returns index of first occurrence of a character.

Parameters

<i>str</i>	The string.
<i>ch</i>	The character for which to search.
<i>start</i>	The index of the string at which to start looking. Set this to non-zero to begin searching from a point other than the first character of the string.

Returns

The index of the first occurrence, or -1 if the character was not found.

5.44.3.26 ds_str ds_str_substr_left (ds_str str, const size_t numchars)

Returns a left substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of left characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

5.44.3.27 ds_str ds_str_substr_right (ds_str str, const size_t numchars)

Returns a right substring.

Parameters

<i>str</i>	The string.
<i>numchars</i>	The number of right characters to return. If this is greater than the length of the string, the whole string is returned.

Returns

A new string representing the substring.

5.44.3.28 void ds_str_trim (ds_str str)

Trims leading and trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

5.44.3.29 void ds_str_trim_leading (ds_str str)

Trims leading whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

5.44.3.30 void ds_str_trim_trailing (ds_str str)

Trims trailing whitespace in-place.

Parameters

<i>str</i>	The string.
------------	-------------

5.44.3.31 ds_str ds_str_trunc (ds_str str, const size_t length)

Truncates a string.

Parameters

<i>str</i>	The string.
<i>length</i>	The new length to which to truncate.

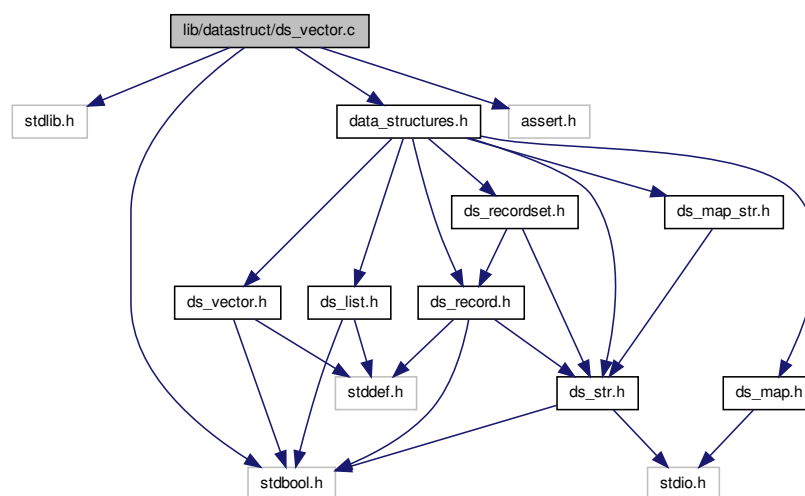
Returns

The original string, or `NULL` on failure.

5.45 lib/datastruct/ds_vector.c File Reference

Implementation of generic doubly-linked vector data structure.

```
#include <stdlib.h>
#include <stdbool.h>
#include <assert.h>
#include "data_structures.h"
Include dependency graph for ds_vector.c:
```



Data Structures

- struct [ds_vector](#)

Functions

- [ds_vector ds_vector_create](#) (const size_t size, const bool free_on_delete, void(*destructor)(void *))
Creates a new vector.
- void [ds_vector_destroy](#) ([ds_vector](#) vector)
Destroys a vector and frees any associated resources.
- void [ds_vector_destructor](#) (void *vector)
A vector destructor function.
- void [ds_vector_clear](#) ([ds_vector](#) vector)
Clears all the elements in a vector.
- void [ds_vector_set](#) ([ds_vector](#) vector, const size_t index, void *element)
Sets an element of a vector.
- void * [ds_vector_element](#) ([ds_vector](#) vector, const size_t index)
Retrieves the data at a specified index.
- size_t [ds_vector_size](#) ([ds_vector](#) vector)
Returns the size of a vector.
- void [ds_vector_seek_start](#) ([ds_vector](#) vector)
Sets the current element to the first element of a vector.
- void * [ds_vector_get_next_data](#) ([ds_vector](#) vector)
Returns the next element of the vector.

5.45.1 Detailed Description

Implementation of generic doubly-linked vector data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.45.2 Function Documentation

5.45.2.1 void ds_vector_clear (ds_vector vector)

Clears all the elements in a vector.

If the vector was created with `free_on_delete`, the elements are `free()`d prior to being cleared (i.e. set to NULL).

Parameters

<i>vector</i>	The vector.
---------------	-------------

5.45.2.2 `ds_vector ds_vector_create (const size_t size, const bool free_on_delete, void(*) (void *) destructor)`

Creates a new vector.

Parameters

<i>size</i>	The size of the vector.
<i>free_on_delete</i>	Set to <code>true</code> if the vector elements should be destroyed when removed from the vector, and when the vector itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the vector.
<i>destructor</i>	Pointer to a destructor function to use for destroying the vector elements, when <code>free_on_delete</code> is true. If this is set to <code>NULL</code> , <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created vector, or `NULL` on failure.

5.45.2.3 `void ds_vector_destroy (ds_vector vector)`

Destroys a vector and frees any associated resources.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

5.45.2.4 `void ds_vector_destructor (void * vector)`

A vector destructor function.

This function may be passed to `ds_vector_create()` when creating a vector of vectors. It calls `ds_vector_destroy()`, but the parameter of `ds_vector_destroy()` is not compatible with the function signature expected by `ds_vector_create()`, so this function provides an appropriate interface.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

5.45.2.5 `void* ds_vector_element (ds_vector vector, const size_t index)`

Retrieves the data at a specified index.

Parameters

<i>vector</i>	The vector from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

5.45.2.6 `void* ds_vector_get_next_data (ds_vector vector)`

Returns the next element of the vector.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

A pointer to the next element, or `NULL` if the end of the vector has been reached.

5.45.2.7 void ds_vector_seek_start (ds_vector vector)

Sets the current element to the first element of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

5.45.2.8 void ds_vector_set (ds_vector vector, const size_t index, void * element)

Sets an element of a vector.

If the element is currently occupied, the existing element is `free()`d.

Parameters

<i>vector</i>	The vector to which to set.
<i>index</i>	The index of the element to set.
<i>element</i>	The element to set.

5.45.2.9 size_t ds_vector_size (ds_vector vector)

Returns the size of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

The size of the vector.

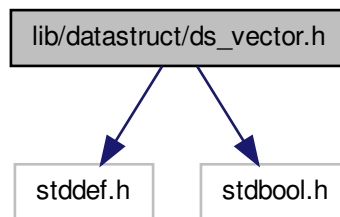
5.46 lib/datastruct/ds_vector.h File Reference

Interface to generic doubly-linked vector data structure.

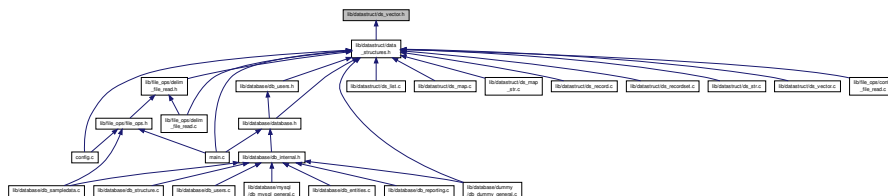
```
#include <stddef.h>
```

```
#include <stdbool.h>
```

Include dependency graph for ds_vector.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct [ds_vector](#) * [ds_vector](#)

Functions

- [ds_vector ds_vector_create](#) (const [size_t](#) size, const bool free_on_delete, void(*destructor)(void *))
Creates a new vector.
- void [ds_vector_destroy](#) ([ds_vector](#) vector)
Destroys a vector and frees any associated resources.
- void [ds_vector_destructor](#) (void *vector)
A vector destructor function.
- void [ds_vector_clear](#) ([ds_vector](#) vector)
Clears all the elements in a vector.
- void [ds_vector_set](#) ([ds_vector](#) vector, const [size_t](#) index, void *element)
Sets an element of a vector.

- void * `ds_vector_element` (`ds_vector` vector, const size_t index)
Retrieves the data at a specified index.
- size_t `ds_vector_size` (`ds_vector` vector)
Returns the size of a vector.
- void `ds_vector_seek_start` (`ds_vector` vector)
Sets the current element to the first element of a vector.
- void * `ds_vector_get_next_data` (`ds_vector` vector)
Returns the next element of the vector.

5.46.1 Detailed Description

Interface to generic doubly-linked vector data structure.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.46.2 Typedef Documentation

5.46.2.1 typedef struct ds_vector* ds_vector

Typedef for opaque vector datatype

5.46.3 Function Documentation

5.46.3.1 void ds_vector_clear (ds_vector vector)

Clears all the elements in a vector.

If the vector was created with `free_on_delete`, the elements are `free()`d prior to being cleared (i.e. set to NULL).

Parameters

<i>vector</i>	The vector.
---------------	-------------

5.46.3.2 ds_vector ds_vector_create (const size_t size, const bool free_on_delete, void(*) (void *) destructor)

Creates a new vector.

Parameters

<i>size</i>	The size of the vector.
<i>free_on_delete</i>	Set to <code>true</code> if the vector elements should be destroyed when removed from the vector, and when the vector itself is destroyed. If set to <code>false</code> , the caller is responsible for destroying the elements prior to destroying the vector.
<i>destructor</i>	Pointer to a destructor function to use for destroying the vector elements, when <code>free_on_delete</code> is true. If this is set to NULL, <code>free()</code> from the standard C library will be used to destroy the elements.

Returns

A newly created vector, or `NULL` on failure.

5.46.3.3 void ds_vector_destroy (ds_vector vector)

Destroys a vector and frees any associated resources.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

5.46.3.4 void ds_vector_destructor (void * vector)

A vector destructor function.

This function may be passed to `ds_vector_create()` when creating a vector of vectors. It calls `ds_vector_destroy()`, but the parameter of `ds_vector_destroy()` is not compatible with the function signature expected by `ds_vector_create()`, so this function provides an appropriate interface.

Parameters

<i>vector</i>	The vector to destroy.
---------------	------------------------

5.46.3.5 void* ds_vector_element (ds_vector vector, const size_t index)

Retrieves the data at a specified index.

Parameters

<i>vector</i>	The vector from which to retrieve.
<i>index</i>	The index of the desired element.

Returns

A pointer to the data, or `NULL` if the index is out of range.

5.46.3.6 void* ds_vector_get_next_data (ds_vector vector)

Returns the next element of the vector.

This function returns the data of the "current element", and advances the current element pointer. Subsequent calls to this function will return successive elements.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

A pointer to the next element, or `NULL` if the end of the vector has been reached.

5.46.3.7 void ds_vector_seek_start (ds_vector vector)

Sets the current element to the first element of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

5.46.3.8 void ds_vector_set (ds_vector vector, const size_t index, void * element)

Sets an element of a vector.

If the element is currently occupied, the existing element is `free()`d.

Parameters

<i>vector</i>	The vector to which to set.
<i>index</i>	The index of the element to set.
<i>element</i>	The element to set.

5.46.3.9 size_t ds_vector_size (ds_vector vector)

Returns the size of a vector.

Parameters

<i>vector</i>	The vector.
---------------	-------------

Returns

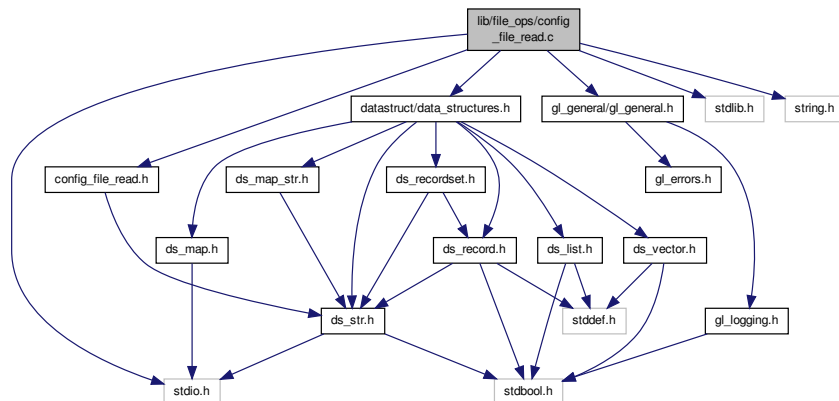
The size of the vector.

5.47 lib/file_ops/config_file_read.c File Reference

Implementation of configuration file reading functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "datastruct/data_structures.h"
#include "config_file_read.h"
```

Include dependency graph for config_file_read.c:



Macros

- #define `MAX_BUFFER_SIZE` 1024
- #define `CONFIG_MAP_SIZE` 100

Functions

- int `config_file_read` (const char *filename)
Reads a configuration file and stores the key-value pairs.
- `ds_str config_file_value` (ds_str key)
Returns the value associated with a key.
- void `config_file_free` (void)
Frees the resources used by this module.

5.47.1 Detailed Description

Implementation of configuration file reading functionality. This module reads configuration files in the format "key = value" and makes those values available. Leading and trailing whitespace is removed for both the key and the value. Blank lines and lines starting with a '#' are ignored in the configuration file.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.47.2 Macro Definition Documentation

5.47.2.1 #define `CONFIG_MAP_SIZE` 100

Size to use for the hash map to contain the key-value pairs

5.47.2.2 `#define MAX_BUFFER_SIZE 1024`

Maximum size of buffers

5.47.3 Function Documentation

5.47.3.1 `void config_file_free (void)`

Frees the resources used by this module.

The user should make copies of any required keys or values prior to calling this function. This function need not be called if `config_file_read()` returned an error.

5.47.3.2 `int config_file_read (const char * filename)`

Reads a configuration file and stores the key-value pairs.

Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

Returns

CONFIG_FILE_OK on success, CONFIG_FILE_NO_FILE if the specified file could not be opened for reading, CONFIG_FILE_MALFORMED_FILE if the configuration file was improperly formed.

5.47.3.3 `ds_str config_file_value (ds_str key)`

Returns the value associated with a key.

Parameters

<i>key</i>	The specified key.
------------	--------------------

Returns

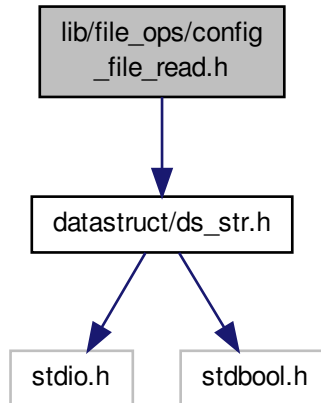
A pointer to the associated value, or `NULL` if the key was not present in the configuration file. The caller should not modify the string to which the pointer points.

5.48 `lib/file_ops/config_file_read.h` File Reference

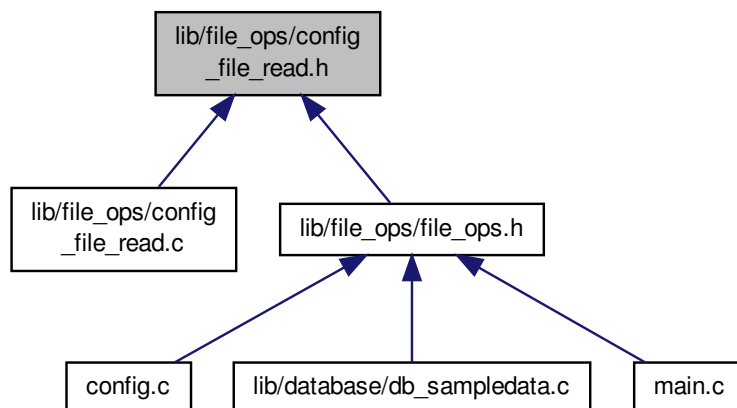
Interface to configuration file reading functionality.


```
#include "datastruct/ds_str.h"
```

Include dependency graph for config_file_read.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define CONFIG_FILE_OK 0`
- `#define CONFIG_FILE_NO_FILE 1`
- `#define CONFIG_FILE_MALFORMED_FILE 2`

Functions

- `int config_file_read (const char *filename)`

Reads a configuration file and stores the key-value pairs.

- void `config_file_free` (void)

Frees the resources used by this module.

- `ds_str config_file_value` (`ds_str` key)

Returns the value associated with a key.

5.48.1 Detailed Description

Interface to configuration file reading functionality. This module reads configuration files in the format "key = value" and makes those values available. Leading and trailing whitespace is removed for both the key and the value. Blank lines and lines starting with a '#' are ignored in the configuration file.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.48.2 Macro Definition Documentation

5.48.2.1 #define CONFIG_FILE_MALFORMED_FILE 2

Return status when configuration file is improperly formed

5.48.2.2 #define CONFIG_FILE_NO_FILE 1

Return status when unable to open file for reading

5.48.2.3 #define CONFIG_FILE_OK 0

Return status for success

5.48.3 Function Documentation

5.48.3.1 void config_file_free (void)

Frees the resources used by this module.

The user should make copies of any required keys or values prior to calling this function. This function need not be called if `config_file_read()` returned an error.

5.48.3.2 int config_file_read (const char * filename)

Reads a configuration file and stores the key-value pairs.

Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

Returns

CONFIG_FILE_OK on success, CONFIG_FILE_NO_FILE if the specified file could not be opened for reading, CONFIG_FILE_MALFORMED_FILE if the configuration file was improperly formed.

5.48.3.3 ds_str config_file_value (ds_str key)

Returns the value associated with a key.

Parameters

<i>key</i>	The specified key.
------------	--------------------

Returns

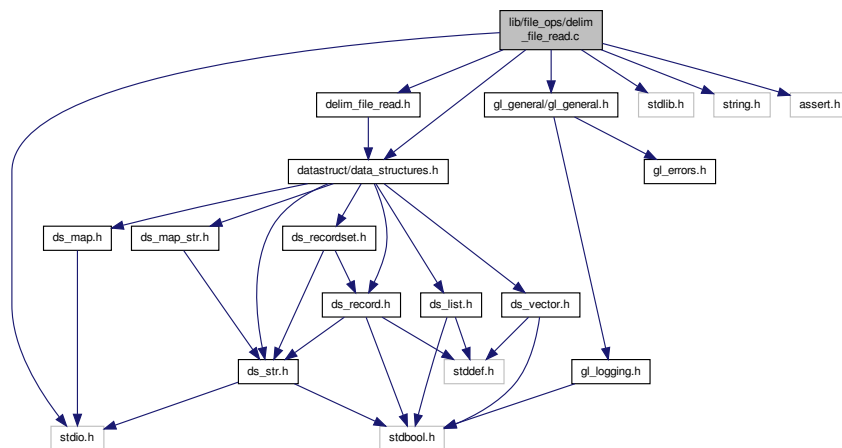
A pointer to the associated value, or NULL if the key was not present in the configuration file. The caller should not modify the string to which the pointer points.

5.49 lib/file_ops/delim_file_read.c File Reference

Implementation of delimited file reading functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "gl_general/gl_general.h"
#include "datastruct/data_structures.h"
#include "delim_file_read.h"
```

Include dependency graph for delim_file_read.c:

**Macros**

- #define MAX_LINE_SIZE 1024

Functions

- [ds_recordset delim_file_read](#) (const char *filename, const char delim)
Constructs a [ds_recordset](#) from a delimited file.

5.49.1 Detailed Description

Implementation of delimited file reading functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.49.2 Macro Definition Documentation

5.49.2.1 #define MAX_LINE_SIZE 1024

Maximum size of buffers

5.49.3 Function Documentation

5.49.3.1 ds_recordset delim_file_read (const char * filename, const char delim)

Constructs a [ds_recordset](#) from a delimited file.

Parameters

<i>filename</i>	The name of the delimited file.
<i>delim</i>	The delimiting character.

Returns

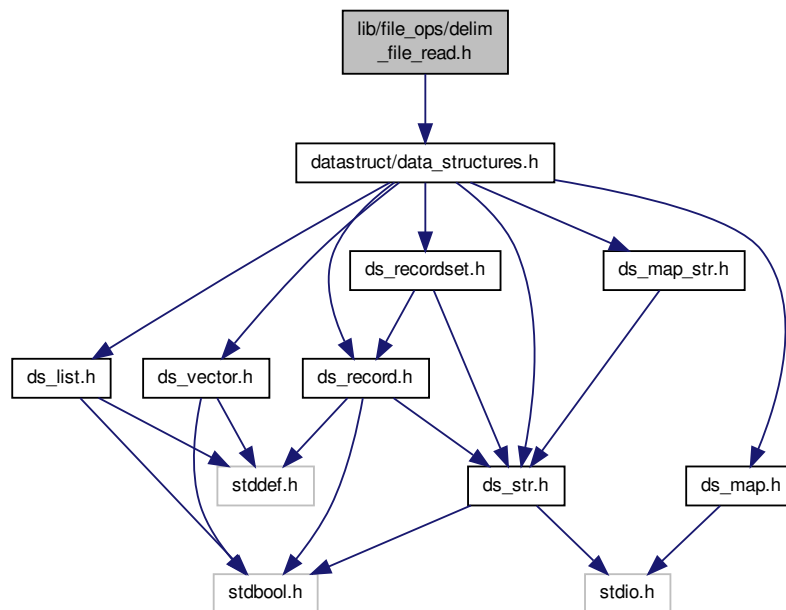
The [ds_recordset](#), or `NULL` on failure.

5.50 lib/file_ops/delim_file_read.h File Reference

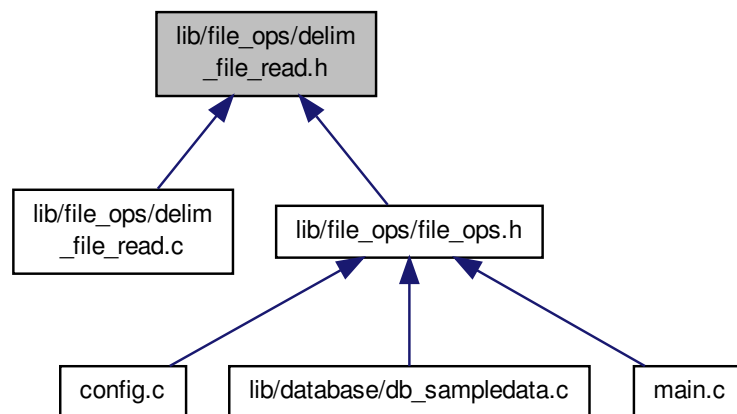
Interface to delimited file reading functionality.

```
#include "datastruct/data_structures.h"
```

Include dependency graph for `delim_file_read.h`:



This graph shows which files directly or indirectly include this file:



Functions

- [`ds_recordset delim_file_read`](#) (const char *filename, const char delim)

Constructs a [`ds_recordset`](#) from a delimited file.

5.50.1 Detailed Description

Interface to delimited file reading functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.50.2 Function Documentation

5.50.2.1 `ds_recordset delim_file_read (const char * filename, const char delim)`

Constructs a `ds_recordset` from a delimited file.

Parameters

<i>filename</i>	The name of the delimited file.
<i>delim</i>	The delimiting character.

Returns

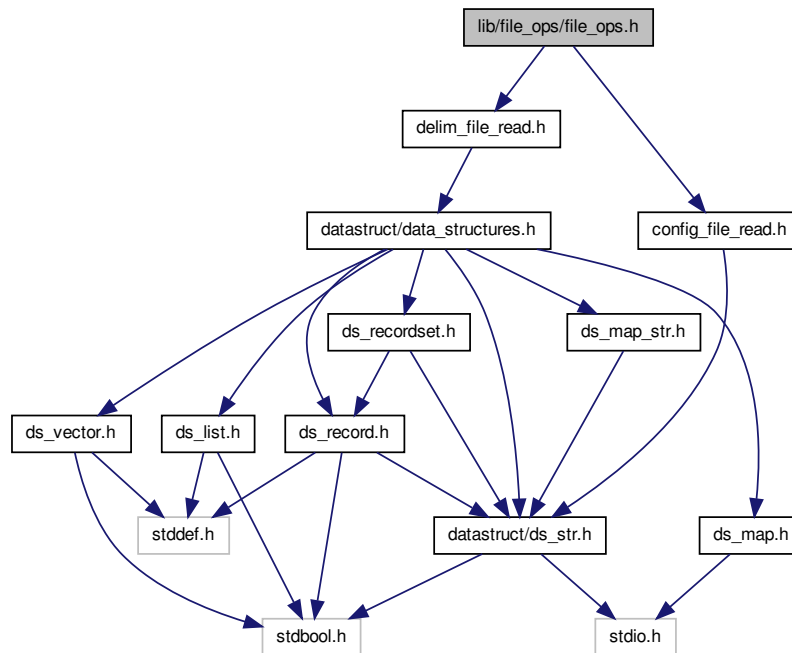
The `ds_recordset`, or `NULL` on failure.

5.51 `lib/file_ops/file_ops.h` File Reference

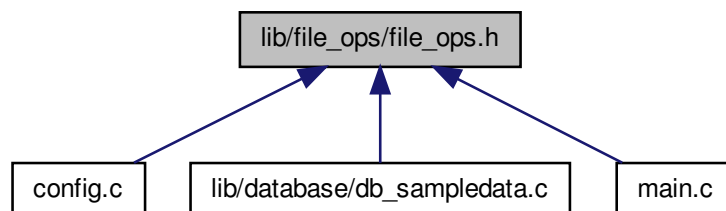
User interface to file operations functionality.

```
#include "config_file_read.h"
#include "delim_file_read.h"
```

Include dependency graph for file_ops.h:



This graph shows which files directly or indirectly include this file:



5.51.1 Detailed Description

User interface to file operations functionality.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

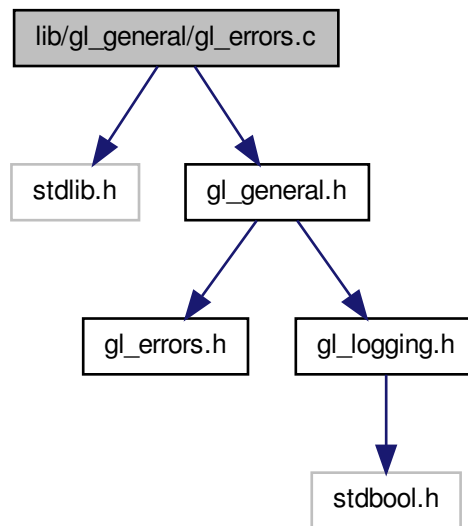
5.52 lib/gl_general/gl_errors.c File Reference

Implementation of error functionality.

```
#include <stdlib.h>
```

```
#include "gl_general.h"
```

Include dependency graph for gl_errors.c:



Functions

- void `gl_error_quit` (const char *msg)
Logs an error message and quits program.

5.52.1 Detailed Description

Implementation of error functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.52.2 Function Documentation

5.52.2.1 void `gl_error_quit` (const char * msg)

Logs an error message and quits program.

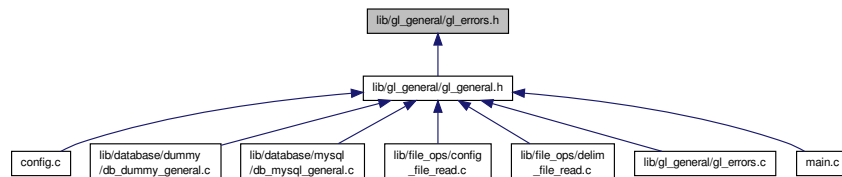
Parameters

<i>msg</i>	The error message to log.
------------	---------------------------

5.53 lib/gl_general/gl_errors.h File Reference

Interface to error functionality.

This graph shows which files directly or indirectly include this file:



Functions

- void [gl_error_quit](#) (const char *msg)
Logs an error message and quits program.

5.53.1 Detailed Description

Interface to error functionality.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.53.2 Function Documentation

5.53.2.1 void gl_error_quit (const char * msg)

Logs an error message and quits program.

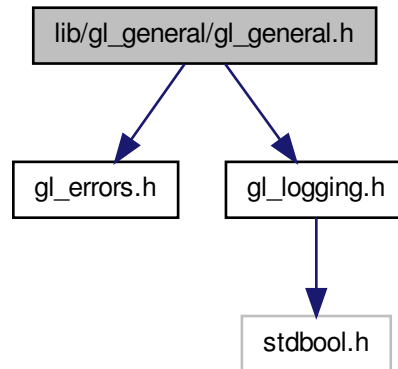
Parameters

<i>msg</i>	The error message to log.
------------	---------------------------

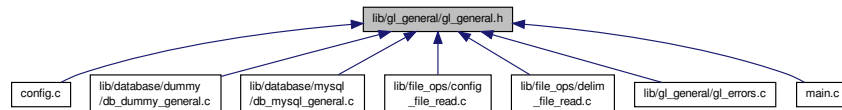
5.54 lib/gl_general/gl_general.h File Reference

User interface to logging and error functionality.

```
#include "gl_errors.h"
#include "gl_logging.h"
Include dependency graph for gl_general.h:
```



This graph shows which files directly or indirectly include this file:



5.54.1 Detailed Description

User interface to logging and error functionality.

Author

Paul Griffiths

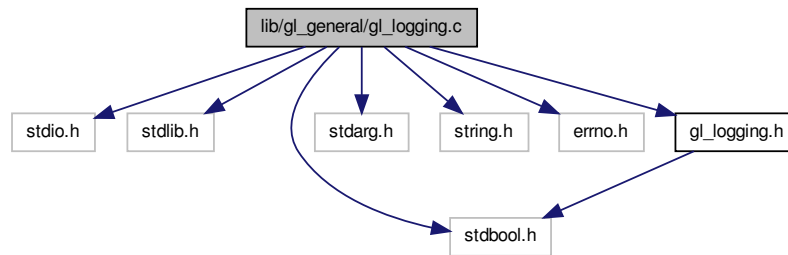
Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.55 lib/gl_general/gl_logging.c File Reference

Implementation of logging functionality.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include "gl_logging.h"
Include dependency graph for gl_logging.c:
```



Functions

- void [gl_set_logging](#) (const bool status)
Turns logging on or off.
- void [gl_log_msg](#) (const char *format,...)
Logs a message to the log file.

5.55.1 Detailed Description

Implementation of logging functionality. Implementation of logging functionality. Enables debugging and other system messages to be recorded to a log file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.55.2 Function Documentation

5.55.2.1 void gl_log_msg (const char * format, ...)

Logs a message to the log file.

Logs a message to the log file.

Parameters

<i>format</i>	Format string, in same format as <code>printf()</code> .
<i>...</i>	Variable arguments as specified by format string.

5.55.2.2 void gl_set_logging (const bool status)

Turns logging on or off.

Turns logging on or off.

Parameters

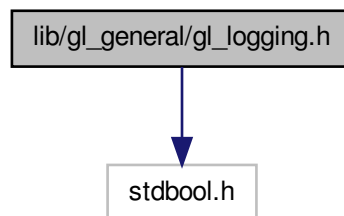
<i>status</i>	true to turn logging on, false to turn logging off.
---------------	---

5.56 lib/gl_general/gl_logging.h File Reference

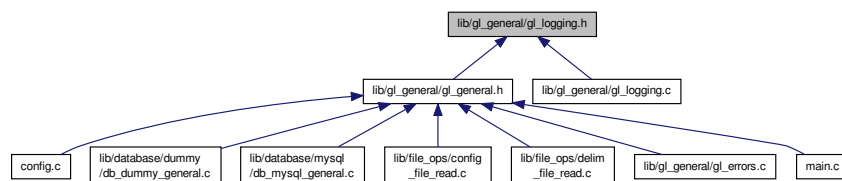
Interface to logging functionality.

```
#include <stdbool.h>
```

Include dependency graph for gl_logging.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [gl_set_logging](#) (const bool status)
Turns logging on or off.
- void [gl_log_msg](#) (const char *format,...)
Logs a message to the log file.

5.56.1 Detailed Description

Interface to logging functionality. Interface to logging functionality. Enables debugging and other system messages to be recorded to a log file.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.56.2 Function Documentation**5.56.2.1 void gl_log_msg (const char * *format*, ...)**

Logs a message to the log file.

Logs a message to the log file.

Parameters

<i>format</i>	Format string, in same format as <code>printf()</code> .
...	Variable arguments as specified by format string.

5.56.2.2 void gl_set_logging (const bool *status*)

Turns logging on or off.

Turns logging on or off.

Parameters

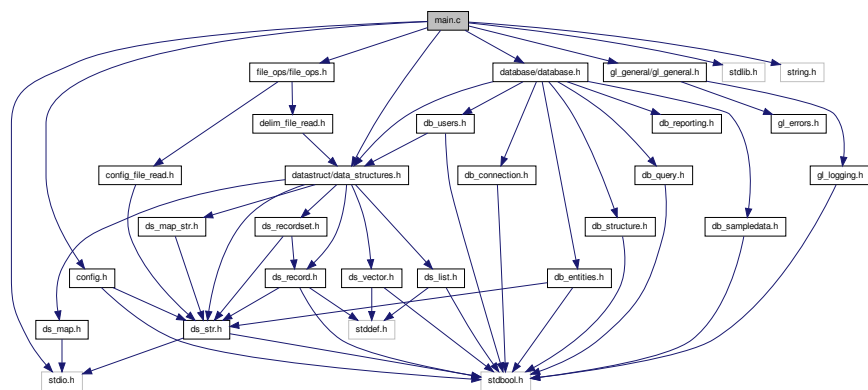
<i>status</i>	true to turn logging on, false to turn logging off.
---------------	---

5.57 main.c File Reference

Main function for `general_ledger`.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gl_general/gl_general.h"
#include "database/database.h"
#include "config.h"
#include "datastruct/data_structures.h"
#include "file_ops/file_ops.h"
```

Include dependency graph for main.c:



Functions

- `ds_str login` (void)
Logs a user in and retrieves the password.
- void `print_usage_message` (char *progname)
Prints a program usage message.
- void `print_version_message` (char *progname)
Prints a program version message.
- void `print_help_message` (char *progname)
Prints a program help message.
- void `test_functionality` (void)
Casual test function.
- int `main` (int argc, char **argv)
Main function.

5.57.1 Detailed Description

Main function for general_ledger. Main function for general_ledger.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.57.2 Function Documentation

5.57.2.1 `ds_str login` (void)

Logs a user in and retrieves the password.

Returns

The password.

5.57.2.2 `int main (int argc, char ** argv)`

Main function.

Main function.

Returns

Exit status.

5.57.2.3 `void print_help_message (char * progrname)`

Prints a program help message.

Parameters

<i>progrname</i>	The program name.
------------------	-------------------

5.57.2.4 `void print_usage_message (char * progrname)`

Prints a program usage message.

Parameters

<i>progrname</i>	The program name.
------------------	-------------------

5.57.2.5 `void print_version_message (char * progrname)`

Prints a program version message.

Parameters

<i>progrname</i>	The program name.
------------------	-------------------

5.57.2.6 `void test_functionality (void)`

Casual test function.

Used for casually testing program functionality.

Index

- `_XOPEN_SOURCE`
 - `config.c`, [22](#)
 - `db_dummy_general.c`, [49](#)
- `CONFIG_FILE_OK`
 - `config_file_read.h`, [122](#)
- `CONFIG_MAP_SIZE`
 - `config_file_read.c`, [119](#)
- `capacity`
 - `ds_str`, [15](#)
- `config.c`, [21](#)
 - `_XOPEN_SOURCE`, [22](#)
 - `get_cmdline_options`, [22](#)
 - `get_configuration`, [22](#)
 - `params_free`, [23](#)
 - `params_init`, [23](#)
- `config.h`, [23](#)
 - `get_cmdline_options`, [24](#)
 - `get_configuration`, [25](#)
 - `params_free`, [25](#)
 - `params_init`, [25](#)
- `config_file_free`
 - `config_file_read.c`, [120](#)
 - `config_file_read.h`, [122](#)
- `config_file_read`
 - `config_file_read.c`, [120](#)
 - `config_file_read.h`, [122](#)
- `config_file_read.c`
 - `CONFIG_MAP_SIZE`, [119](#)
 - `config_file_free`, [120](#)
 - `config_file_read`, [120](#)
 - `config_file_value`, [120](#)
 - `MAX_BUFFER_SIZE`, [119](#)
- `config_file_read.h`
 - `CONFIG_FILE_OK`, [122](#)
 - `config_file_free`, [122](#)
 - `config_file_read`, [122](#)
 - `config_file_value`, [123](#)
- `config_file_value`
 - `config_file_read.c`, [120](#)
 - `config_file_read.h`, [123](#)
- `conn_mss`
 - `db_mysql_general.c`, [55](#)
- `create`
 - `params`, [18](#)
- `current`
 - `ds_list`, [9](#)
 - `ds_vector`, [16](#)
- `data`
 - `ds_list_element`, [11](#)
 - `ds_str`, [15](#)
 - `ds_vector`, [16](#)
- `data_destructor`
 - `ds_list`, [10](#)
 - `ds_vector`, [16](#)
- `database`
 - `params`, [18](#)
- `db_connect`
 - `db_connection.h`, [28](#)
 - `db_dummy_general.c`, [49](#)
 - `db_mysql_general.c`, [55](#)
- `db_connection.h`
 - `db_connect`, [28](#)
- `db_create_database_structure`
 - `db_structure.c`, [40](#)
 - `db_structure.h`, [42](#)
- `db_create_entities_table`
 - `db_entities.c`, [29](#)
 - `db_entities.h`, [31](#)
- `db_create_entities_table_sql`
 - `db_dummy_create_entities_table_sql.c`, [46](#)
 - `db_mysql_create_entities_table_sql.c`, [51](#)
 - `db_sql.h`, [38](#)
- `db_create_recordset_from_query`
 - `db_dummy_general.c`, [49](#)
 - `db_mysql_general.c`, [55](#)
 - `db_reporting.h`, [35](#)
- `db_create_report_from_query`
 - `db_reporting.c`, [34](#)
 - `db_reporting.h`, [35](#)
- `db_create_users_table`
 - `db_users.c`, [43](#)
 - `db_users.h`, [45](#)
- `db_create_users_table_sql`
 - `db_dummy_create_users_table_sql.c`, [46](#)
 - `db_mysql_create_users_table_sql.c`, [52](#)
 - `db_sql.h`, [38](#)
- `db_delete_database_structure`
 - `db_structure.c`, [40](#)
 - `db_structure.h`, [42](#)
- `db_drop_entities_table`
 - `db_entities.c`, [29](#)
 - `db_entities.h`, [31](#)
- `db_drop_entities_table_sql`
 - `db_dummy_drop_entities_table_sql.c`, [47](#)
 - `db_mysql_drop_entities_table_sql.c`, [53](#)
 - `db_sql.h`, [39](#)
- `db_drop_users_table`

- db_users.c, 43
- db_users.h, 45
- db_drop_users_table_sql
 - db_dummy_drop_users_table_sql.c, 47
 - db_mysql_drop_users_table_sql.c, 53
 - db_sql.h, 39
- db_dummy_create_entities_table_sql.c
 - db_create_entities_table_sql, 46
- db_dummy_create_users_table_sql.c
 - db_create_users_table_sql, 46
- db_dummy_drop_entities_table_sql.c
 - db_drop_entities_table_sql, 47
- db_dummy_drop_users_table_sql.c
 - db_drop_users_table_sql, 47
- db_dummy_general.c
 - _XOPEN_SOURCE, 49
 - db_connect, 49
 - db_create_recordset_from_query, 49
 - db_execute_query, 49
- db_dummy_list_entities_report_sql.c
 - db_list_entities_report_sql, 50
- db_dummy_list_users_report_sql.c
 - db_list_users_report_sql, 51
- db_entities.c
 - db_create_entities_table, 29
 - db_drop_entities_table, 29
 - db_list_entities_report, 29
- db_entities.h
 - db_create_entities_table, 31
 - db_drop_entities_table, 31
 - db_list_entities_report, 31
- db_execute_query
 - db_dummy_general.c, 49
 - db_mysql_general.c, 55
 - db_query.h, 33
- db_list_entities_report
 - db_entities.c, 29
 - db_entities.h, 31
- db_list_entities_report_sql
 - db_dummy_list_entities_report_sql.c, 50
 - db_mysql_list_entities_report_sql.c, 56
 - db_sql.h, 39
- db_list_users_report
 - db_users.c, 43
 - db_users.h, 45
- db_list_users_report_sql
 - db_dummy_list_users_report_sql.c, 51
 - db_mysql_list_users_report_sql.c, 57
 - db_sql.h, 39
- db_mysql_create_entities_table_sql.c
 - db_create_entities_table_sql, 51
- db_mysql_create_users_table_sql.c
 - db_create_users_table_sql, 52
- db_mysql_drop_entities_table_sql.c
 - db_drop_entities_table_sql, 53
- db_mysql_drop_users_table_sql.c
 - db_drop_users_table_sql, 53
- db_mysql_general.c
 - conn_mss, 55
 - db_connect, 55
 - db_create_recordset_from_query, 55
 - db_execute_query, 55
 - main_mss, 55
- db_mysql_list_entities_report_sql.c
 - db_list_entities_report_sql, 56
- db_mysql_list_users_report_sql.c
 - db_list_users_report_sql, 57
- db_query.h
 - db_execute_query, 33
- db_reporting.c
 - db_create_report_from_query, 34
- db_reporting.h
 - db_create_recordset_from_query, 35
 - db_create_report_from_query, 35
- db_sql.h
 - db_create_entities_table_sql, 38
 - db_create_users_table_sql, 38
 - db_drop_entities_table_sql, 39
 - db_drop_users_table_sql, 39
 - db_list_entities_report_sql, 39
 - db_list_users_report_sql, 39
- db_structure.c
 - db_create_database_structure, 40
 - db_delete_database_structure, 40
- db_structure.h
 - db_create_database_structure, 42
 - db_delete_database_structure, 42
- db_users.c
 - db_create_users_table, 43
 - db_drop_users_table, 43
 - db_list_users_report, 43
- db_users.h
 - db_create_users_table, 45
 - db_drop_users_table, 45
 - db_list_users_report, 45
- delete_data
 - params, 18
- delim_file_read
 - delim_file_read.c, 124
 - delim_file_read.h, 126
- delim_file_read.c
 - delim_file_read, 124
 - MAX_LINE_SIZE, 124
- delim_file_read.h
 - delim_file_read, 126
- ds_list, 9
 - current, 9
 - data_destructor, 10
 - ds_list.h, 64
 - free_on_delete, 10
 - head, 10
 - length, 10
 - tail, 10
- ds_list.c
 - ds_list_append, 59
 - ds_list_create, 59

- ds_list_destroy, 60
- ds_list_destructor, 60
- ds_list_element, 60
- ds_list_get_next_data, 60
- ds_list_get_prev_data, 61
- ds_list_is_empty, 61
- ds_list_length, 61
- ds_list_remove_all, 61
- ds_list_remove_tail, 62
- ds_list_seek_end, 62
- ds_list_seek_start, 62
- ds_list.h
 - ds_list, 64
 - ds_list_append, 64
 - ds_list_create, 64
 - ds_list_destroy, 64
 - ds_list_destructor, 64
 - ds_list_element, 65
 - ds_list_get_next_data, 65
 - ds_list_get_prev_data, 65
 - ds_list_is_empty, 65
 - ds_list_length, 66
 - ds_list_remove_all, 66
 - ds_list_remove_tail, 66
 - ds_list_seek_end, 66
 - ds_list_seek_start, 66
- ds_list_append
 - ds_list.c, 59
 - ds_list.h, 64
- ds_list_create
 - ds_list.c, 59
 - ds_list.h, 64
- ds_list_destroy
 - ds_list.c, 60
 - ds_list.h, 64
- ds_list_destructor
 - ds_list.c, 60
 - ds_list.h, 64
- ds_list_element, 10
 - data, 11
 - ds_list.c, 60
 - ds_list.h, 65
 - next, 11
 - previous, 11
- ds_list_get_next_data
 - ds_list.c, 60
 - ds_list.h, 65
- ds_list_get_prev_data
 - ds_list.c, 61
 - ds_list.h, 65
- ds_list_is_empty
 - ds_list.c, 61
 - ds_list.h, 65
- ds_list_length
 - ds_list.c, 61
 - ds_list.h, 66
- ds_list_remove_all
 - ds_list.c, 61
- ds_list.h, 66
- ds_list_remove_tail
 - ds_list.c, 62
 - ds_list.h, 66
- ds_list_seek_end
 - ds_list.c, 62
 - ds_list.h, 66
- ds_list_seek_start
 - ds_list.c, 62
 - ds_list.h, 66
- ds_map, 11
 - ds_map.h, 70
 - hash_size, 12
 - lists, 12
- ds_map.c
 - ds_map_destroy, 68
 - ds_map_get_value, 68
 - ds_map_init, 68
 - ds_map_insert, 69
 - ds_map_print_all, 69
- ds_map.h
 - ds_map, 70
 - ds_map_destroy, 70
 - ds_map_get_value, 71
 - ds_map_init, 71
 - ds_map_insert, 71
 - ds_map_print_all, 71
- ds_map_destroy
 - ds_map.c, 68
 - ds_map.h, 70
- ds_map_get_value
 - ds_map.c, 68
 - ds_map.h, 71
- ds_map_init
 - ds_map.c, 68
 - ds_map.h, 71
- ds_map_insert
 - ds_map.c, 69
 - ds_map.h, 71
- ds_map_print_all
 - ds_map.c, 69
 - ds_map.h, 71
- ds_map_str, 12
 - ds_map_str.h, 75
 - hash_size, 12
 - lists, 13
- ds_map_str.c
 - ds_map_str_destroy, 73
 - ds_map_str_get_value, 73
 - ds_map_str_init, 73
 - ds_map_str_insert, 73
- ds_map_str.h
 - ds_map_str, 75
 - ds_map_str_destroy, 75
 - ds_map_str_get_value, 75
 - ds_map_str_init, 75
 - ds_map_str_insert, 76
- ds_map_str_destroy

- ds_map_str.c, 73
 - ds_map_str.h, 75
- ds_map_str_get_value
 - ds_map_str.c, 73
 - ds_map_str.h, 75
- ds_map_str_init
 - ds_map_str.c, 73
 - ds_map_str.h, 75
- ds_map_str_insert
 - ds_map_str.c, 73
 - ds_map_str.h, 76
- ds_record, 13
 - ds_record.h, 81
 - fields, 13
- ds_record.c
 - ds_record_clear, 77
 - ds_record_create, 77
 - ds_record_destroy, 78
 - ds_record_destructor, 78
 - ds_record_get_field, 78
 - ds_record_get_next_data, 78
 - ds_record_make_delim_string, 78
 - ds_record_make_values_string, 79
 - ds_record_seek_start, 79
 - ds_record_set_field, 79
 - ds_record_size, 79
 - ds_record_tokenize, 79
- ds_record.h
 - ds_record, 81
 - ds_record_clear, 81
 - ds_record_create, 82
 - ds_record_destroy, 82
 - ds_record_destructor, 82
 - ds_record_get_field, 82
 - ds_record_get_next_data, 82
 - ds_record_make_delim_string, 83
 - ds_record_make_values_string, 83
 - ds_record_seek_start, 83
 - ds_record_set_field, 83
 - ds_record_size, 83
 - ds_record_tokenize, 84
- ds_record_clear
 - ds_record.c, 77
 - ds_record.h, 81
- ds_record_create
 - ds_record.c, 77
 - ds_record.h, 82
- ds_record_destroy
 - ds_record.c, 78
 - ds_record.h, 82
- ds_record_destructor
 - ds_record.c, 78
 - ds_record.h, 82
- ds_record_get_field
 - ds_record.c, 78
 - ds_record.h, 82
- ds_record_get_next_data
 - ds_record.c, 78
- ds_record.h, 82
 - ds_record, 82
- ds_record_make_delim_string
 - ds_record.c, 78
 - ds_record.h, 83
- ds_record_make_values_string
 - ds_record.c, 79
 - ds_record.h, 83
- ds_record_seek_start
 - ds_record.c, 79
 - ds_record.h, 83
- ds_record_set_field
 - ds_record.c, 79
 - ds_record.h, 83
- ds_record_size
 - ds_record.c, 79
 - ds_record.h, 83
- ds_record_tokenize
 - ds_record.c, 79
 - ds_record.h, 84
- ds_recordset, 14
 - ds_recordset.h, 89
 - field_lengths, 14
 - headers, 14
 - num_fields, 14
 - records, 14
- ds_recordset.c
 - ds_recordset_add_record, 85
 - ds_recordset_create, 86
 - ds_recordset_destroy, 86
 - ds_recordset_get_next_insert_query, 86
 - ds_recordset_get_text_report, 86
 - ds_recordset_next_record, 86
 - ds_recordset_num_fields, 87
 - ds_recordset_num_records, 87
 - ds_recordset_seek_start, 87
 - ds_recordset_set_headers, 87
- ds_recordset.h
 - ds_recordset, 89
 - ds_recordset_add_record, 89
 - ds_recordset_create, 89
 - ds_recordset_destroy, 90
 - ds_recordset_get_next_insert_query, 90
 - ds_recordset_get_text_report, 90
 - ds_recordset_next_record, 90
 - ds_recordset_num_fields, 91
 - ds_recordset_num_records, 91
 - ds_recordset_seek_start, 91
 - ds_recordset_set_headers, 91
- ds_recordset_add_record
 - ds_recordset.c, 85
 - ds_recordset.h, 89
- ds_recordset_create
 - ds_recordset.c, 86
 - ds_recordset.h, 89
- ds_recordset_destroy
 - ds_recordset.c, 86
 - ds_recordset.h, 90
- ds_recordset_get_next_insert_query

- ds_recordset.c, 86
 - ds_recordset.h, 90
- ds_recordset_get_text_report
 - ds_recordset.c, 86
 - ds_recordset.h, 90
- ds_recordset_next_record
 - ds_recordset.c, 86
 - ds_recordset.h, 90
- ds_recordset_num_fields
 - ds_recordset.c, 87
 - ds_recordset.h, 91
- ds_recordset_num_records
 - ds_recordset.c, 87
 - ds_recordset.h, 91
- ds_recordset_seek_start
 - ds_recordset.c, 87
 - ds_recordset.h, 91
- ds_recordset_set_headers
 - ds_recordset.c, 87
 - ds_recordset.h, 91
- ds_str, 15
 - capacity, 15
 - data, 15
 - ds_str.h, 103
 - length, 15
- ds_str.c
 - ds_str_assign, 94
 - ds_str_assign_cstr, 94
 - ds_str_char_at_index, 94
 - ds_str_clear, 94
 - ds_str_compare, 95
 - ds_str_compare_cstr, 95
 - ds_str_concat, 95
 - ds_str_concat_cstr, 95
 - ds_str_create, 96
 - ds_str_create_direct, 96
 - ds_str_create_sprintf, 96
 - ds_str_cstr, 96
 - ds_str_decorate, 97
 - ds_str_destroy, 97
 - ds_str_destructor, 97
 - ds_str_doubleval, 97
 - ds_str_dup, 98
 - ds_str_getline, 98
 - ds_str_hash, 98
 - ds_str_intval, 98
 - ds_str_is_empty, 99
 - ds_str_length, 99
 - ds_str_size_to_fit, 99
 - ds_str_split, 99
 - ds_str_strchr, 100
 - ds_str_substr_left, 100
 - ds_str_substr_right, 100
 - ds_str_trim, 100
 - ds_str_trim_leading, 100
 - ds_str_trim_trailing, 101
 - ds_str_trunc, 101
- ds_str.h
 - ds_str, 103
 - ds_str_assign, 104
 - ds_str_assign_cstr, 104
 - ds_str_char_at_index, 104
 - ds_str_clear, 104
 - ds_str_compare, 104
 - ds_str_compare_cstr, 105
 - ds_str_concat, 105
 - ds_str_concat_cstr, 105
 - ds_str_create, 105
 - ds_str_create_direct, 106
 - ds_str_create_sprintf, 106
 - ds_str_cstr, 106
 - ds_str_decorate, 107
 - ds_str_destroy, 107
 - ds_str_destructor, 107
 - ds_str_doubleval, 107
 - ds_str_dup, 107
 - ds_str_getline, 108
 - ds_str_hash, 108
 - ds_str_intval, 108
 - ds_str_is_empty, 108
 - ds_str_length, 109
 - ds_str_size_to_fit, 109
 - ds_str_split, 109
 - ds_str_strchr, 109
 - ds_str_substr_left, 110
 - ds_str_substr_right, 110
 - ds_str_trim, 110
 - ds_str_trim_leading, 110
 - ds_str_trim_trailing, 111
 - ds_str_trunc, 111
- ds_str_assign
 - ds_str.c, 94
 - ds_str.h, 104
- ds_str_assign_cstr
 - ds_str.c, 94
 - ds_str.h, 104
- ds_str_char_at_index
 - ds_str.c, 94
 - ds_str.h, 104
- ds_str_clear
 - ds_str.c, 94
 - ds_str.h, 104
- ds_str_compare
 - ds_str.c, 95
 - ds_str.h, 104
- ds_str_compare_cstr
 - ds_str.c, 95
 - ds_str.h, 105
- ds_str_concat
 - ds_str.c, 95
 - ds_str.h, 105
- ds_str_concat_cstr
 - ds_str.c, 95
 - ds_str.h, 105
- ds_str_create
 - ds_str.c, 96

- ds_str.h, 105
- ds_str_create_direct
 - ds_str.c, 96
 - ds_str.h, 106
- ds_str_create_sprintf
 - ds_str.c, 96
 - ds_str.h, 106
- ds_str_cstr
 - ds_str.c, 96
 - ds_str.h, 106
- ds_str_decorate
 - ds_str.c, 97
 - ds_str.h, 107
- ds_str_destroy
 - ds_str.c, 97
 - ds_str.h, 107
- ds_str_destructor
 - ds_str.c, 97
 - ds_str.h, 107
- ds_str_doubleval
 - ds_str.c, 97
 - ds_str.h, 107
- ds_str_dup
 - ds_str.c, 98
 - ds_str.h, 107
- ds_str_getline
 - ds_str.c, 98
 - ds_str.h, 108
- ds_str_hash
 - ds_str.c, 98
 - ds_str.h, 108
- ds_str_intval
 - ds_str.c, 98
 - ds_str.h, 108
- ds_str_is_empty
 - ds_str.c, 99
 - ds_str.h, 108
- ds_str_length
 - ds_str.c, 99
 - ds_str.h, 109
- ds_str_size_to_fit
 - ds_str.c, 99
 - ds_str.h, 109
- ds_str_split
 - ds_str.c, 99
 - ds_str.h, 109
- ds_str_strchr
 - ds_str.c, 100
 - ds_str.h, 109
- ds_str_substr_left
 - ds_str.c, 100
 - ds_str.h, 110
- ds_str_substr_right
 - ds_str.c, 100
 - ds_str.h, 110
- ds_str_trim
 - ds_str.c, 100
 - ds_str.h, 110
- ds_str_trim_leading
 - ds_str.c, 100
 - ds_str.h, 110
- ds_str_trim_trailing
 - ds_str.c, 101
 - ds_str.h, 111
- ds_str_trunc
 - ds_str.c, 101
 - ds_str.h, 111
- ds_vector, 15
 - current, 16
 - data, 16
 - data_destructor, 16
 - ds_vector.h, 116
 - free_on_delete, 16
 - size, 16
- ds_vector.c
 - ds_vector_clear, 112
 - ds_vector_create, 112
 - ds_vector_destroy, 113
 - ds_vector_destructor, 113
 - ds_vector_element, 113
 - ds_vector_get_next_data, 113
 - ds_vector_seek_start, 114
 - ds_vector_set, 114
 - ds_vector_size, 114
- ds_vector.h
 - ds_vector, 116
 - ds_vector_clear, 116
 - ds_vector_create, 116
 - ds_vector_destroy, 117
 - ds_vector_destructor, 117
 - ds_vector_element, 117
 - ds_vector_get_next_data, 117
 - ds_vector_seek_start, 118
 - ds_vector_set, 118
 - ds_vector_size, 118
- ds_vector_clear
 - ds_vector.c, 112
 - ds_vector.h, 116
- ds_vector_create
 - ds_vector.c, 112
 - ds_vector.h, 116
- ds_vector_destroy
 - ds_vector.c, 113
 - ds_vector.h, 117
- ds_vector_destructor
 - ds_vector.c, 113
 - ds_vector.h, 117
- ds_vector_element
 - ds_vector.c, 113
 - ds_vector.h, 117
- ds_vector_get_next_data
 - ds_vector.c, 113
 - ds_vector.h, 117
- ds_vector_seek_start
 - ds_vector.c, 114
 - ds_vector.h, 118

- ds_vector_set
 - ds_vector.c, [114](#)
 - ds_vector.h, [118](#)
- ds_vector_size
 - ds_vector.c, [114](#)
 - ds_vector.h, [118](#)
- field_lengths
 - ds_recordset, [14](#)
- fields
 - ds_record, [13](#)
- free_on_delete
 - ds_list, [10](#)
 - ds_vector, [16](#)
- get_cmdline_options
 - config.c, [22](#)
 - config.h, [24](#)
- get_configuration
 - config.c, [22](#)
 - config.h, [25](#)
- gl_error_quit
 - gl_errors.c, [128](#)
 - gl_errors.h, [129](#)
- gl_errors.c
 - gl_error_quit, [128](#)
- gl_errors.h
 - gl_error_quit, [129](#)
- gl_log_msg
 - gl_logging.c, [131](#)
 - gl_logging.h, [133](#)
- gl_logging.c
 - gl_log_msg, [131](#)
 - gl_set_logging, [132](#)
- gl_logging.h
 - gl_log_msg, [133](#)
 - gl_set_logging, [133](#)
- gl_set_logging
 - gl_logging.c, [132](#)
 - gl_logging.h, [133](#)
- hash_size
 - ds_map, [12](#)
 - ds_map_str, [12](#)
- head
 - ds_list, [10](#)
- headers
 - ds_recordset, [14](#)
- help
 - params, [18](#)
- hostname
 - params, [19](#)
- key
 - kv_pair_node, [17](#)
- kv_pair_node, [16](#)
 - key, [17](#)
 - next, [17](#)
 - value, [17](#)
- length
 - ds_list, [10](#)
 - ds_str, [15](#)
- lib/database/database.h, [25](#)
- lib/database/db_connection.h, [27](#)
- lib/database/db_entities.c, [28](#)
- lib/database/db_entities.h, [29](#)
- lib/database/db_internal.h, [31](#)
- lib/database/db_query.h, [32](#)
- lib/database/db_reporting.c, [33](#)
- lib/database/db_reporting.h, [35](#)
- lib/database/db_sampledata.c, [36](#)
- lib/database/db_sampledata.h, [37](#)
- lib/database/db_sql.h, [37](#)
- lib/database/db_structure.c, [39](#)
- lib/database/db_structure.h, [41](#)
- lib/database/db_users.c, [42](#)
- lib/database/db_users.h, [44](#)
- lib/database/dummy/db_dummy_create_entities_table_sql.c, [45](#)
- lib/database/dummy/db_dummy_create_users_table_sql.c, [46](#)
- lib/database/dummy/db_dummy_drop_entities_table_sql.c, [46](#)
- lib/database/dummy/db_dummy_drop_users_table_sql.c, [47](#)
- lib/database/dummy/db_dummy_general.c, [48](#)
- lib/database/dummy/db_dummy_list_entities_report_sql.c, [50](#)
- lib/database/dummy/db_dummy_list_users_report_sql.c, [50](#)
- lib/database/mysql/db_mysql_create_entities_table_sql.c, [51](#)
- lib/database/mysql/db_mysql_create_users_table_sql.c, [51](#)
- lib/database/mysql/db_mysql_drop_entities_table_sql.c, [52](#)
- lib/database/mysql/db_mysql_drop_users_table_sql.c, [53](#)
- lib/database/mysql/db_mysql_general.c, [53](#)
- lib/database/mysql/db_mysql_list_entities_report_sql.c, [56](#)
- lib/database/mysql/db_mysql_list_users_report_sql.c, [56](#)
- lib/datastruct/data_structures.h, [57](#)
- lib/datastruct/ds_list.c, [58](#)
- lib/datastruct/ds_list.h, [62](#)
- lib/datastruct/ds_map.c, [67](#)
- lib/datastruct/ds_map.h, [69](#)
- lib/datastruct/ds_map_str.c, [72](#)
- lib/datastruct/ds_map_str.h, [74](#)
- lib/datastruct/ds_record.c, [76](#)
- lib/datastruct/ds_record.h, [80](#)
- lib/datastruct/ds_recordset.c, [84](#)
- lib/datastruct/ds_recordset.h, [88](#)
- lib/datastruct/ds_str.c, [92](#)
- lib/datastruct/ds_str.h, [101](#)
- lib/datastruct/ds_vector.c, [111](#)

- lib/datastruct/ds_vector.h, 115
- lib/file_ops/config_file_read.c, 118
- lib/file_ops/config_file_read.h, 120
- lib/file_ops/delim_file_read.c, 123
- lib/file_ops/delim_file_read.h, 124
- lib/file_ops/file_ops.h, 126
- lib/gl_general/gl_errors.c, 128
- lib/gl_general/gl_errors.h, 129
- lib/gl_general/gl_general.h, 129
- lib/gl_general/gl_logging.c, 130
- lib/gl_general/gl_logging.h, 132
- list_entities
 - params, 19
- list_users
 - params, 19
- lists
 - ds_map, 12
 - ds_map_str, 13
- login
 - main.c, 134
- MAX_BUFFER_SIZE
 - config_file_read.c, 119
- MAX_LINE_SIZE
 - delim_file_read.c, 124
- main
 - main.c, 134
- main.c, 133
 - login, 134
 - main, 134
 - print_help_message, 135
 - print_usage_message, 135
 - print_version_message, 135
 - test_functionality, 135
- main_mss
 - db_mysql_general.c, 55
- next
 - ds_list_element, 11
 - kv_pair_node, 17
- num_fields
 - ds_recordset, 14
- params, 17
 - create, 18
 - database, 18
 - delete_data, 18
 - help, 18
 - hostname, 19
 - list_entities, 19
 - list_users, 19
 - password, 19
 - sample, 19
 - username, 19
 - version, 19
- params_free
 - config.c, 23
 - config.h, 25
- params_init
 - config.c, 23
 - config.h, 25
- password
 - params, 19
- previous
 - ds_list_element, 11
- print_help_message
 - main.c, 135
- print_usage_message
 - main.c, 135
- print_version_message
 - main.c, 135
- records
 - ds_recordset, 14
- sample
 - params, 19
- size
 - ds_vector, 16
- tail
 - ds_list, 10
- test_functionality
 - main.c, 135
- username
 - params, 19
- value
 - kv_pair_node, 17
- version
 - params, 19